Propósito y Funcionalidad

Visualización SVG interactiva que representa dispositivos físicos como silos con sensores posicionados estratégicamente.

Características Principales

- SVG dinámico: Representación visual del dispositivo
- Sensores interactivos: Tooltips con datos en tiempo real
- Detección de estados: Normal, advertencia, alerta
- Animaciones CSS: Efectos hover y pulsación para alertas

Características Técnicas

Arquitectura SVG

```javascript
      // Sistema de coordenadas viewBox
      const svg = el("svg", {
        viewBox: "0 0 400 600", // Coordenadas abstractas
        style: "width: 100%; height: 400px; max-width: 600px;",
        xmlns: "http://www.w3.org/2000/svg"
      });

      // Elementos estructurales del dispositivo
      const deviceBody = el("ellipse", {
```

```
    cx: "200", cy: "350", rx: "120", ry: "180", // Cuerpo principal
    fill: "#E0E0E0", stroke: "#BDBDBD", "stroke-width": "3"
  });

  const deviceTop = el("path", {
    d: "M80 350 L200 150 L320 350 Z", // Techo triangular
    fill: "#9E9E9E", stroke: "#757575", "stroke-width": "2"
});
```

Sistema de Posicionamiento de Sensores

```
javascript
  function getSensorPosition(index, total) {
    const positions = [
      { x: 200, y: 200, label: 'T' }, // Superior
      { x: 200, y: 300, label: 'H' }, // Medio-superior
      { x: 200, y: 400, label: 'M' }, // Medio
      { x: 200, y: 500, label: 'B' }, // Inferior
      { x: 120, y: 350, label: 'L' }, // Lateral izquierdo
      { x: 280, y: 350, label: 'R' }, // Lateral derecho
      { x: 200, y: 250, label: 'C' }, // Centro-superior
      { x: 200, y: 450, label: 'D' }  // Centro-inferior
    ];
```

```
    return positions[index % positions.length]; // Distribución
  circular
}
```

Algoritmo de Estado de Sensores

```
javascript
  function getSensorStatus(sensor) {
    const value = parseFloat(sensor.valor);
    const thresholds = {
      'temperatura': { min: 0, max: 50 },
      'humedad': { min: 20, max: 80 },
      'co2': { min: 300, max: 1000 }
      // ... más configuraciones
    };

    const threshold = thresholds[sensor.tipo_sensor];
    if (!threshold) return 'normal';

    // Lógica de estados con zonas de advertencia
    if (value < threshold.min || value > threshold.max) {
      return 'alert'; // Crítico
    } else if (value < (threshold.min + (threshold.max - threshold.min)
  * 0.1) ||
```

```
                value > (threshold.max - (threshold.max - threshold.min)
    * 0.1)) {
      return 'warning'; // Advertencia (10% de los límites)
    }
    return 'normal'; // Dentro de rangos seguros
}
```

Sistema de Tooltips Avanzado

```
javascript
    function showTooltip(event, sensor) {
      const tooltipElement = document.getElementById('sensor-tooltip');
      const status = getSensorStatus(sensor);
      const color = getStatusColor(status);
      const statusText = status === 'alert' ? 'Alerta' :
                         status === 'warning' ? 'Advertencia' : 'Normal';

      // Contenido dinámico del tooltip
      tooltipElement.innerHTML = `
        <h3 style="margin-top: 0; margin-bottom: 10px; color: ${color};">
          ${sensor.tipo_sensor.charAt(0).toUpperCase() +
    sensor.tipo_sensor.slice(1)}
        </h3>
        <p style="margin: 5px 0;"><strong>Valor:</strong>
    ${sensor.valor}${sensor.unidad || ''}</p>
```

```
        <p style="margin: 5px 0;"><strong>Estado:</strong>
          <span style="color: ${color}">${statusText}</span>
        </p>
        <p style="margin: 5px 0;"><strong>Última lectura:</strong>
          ${new Date(sensor.timestamp).toLocaleString()}
        </p>
      `;

      // Posicionamiento inteligente
      const rect = event.target.getBoundingClientRect();
      const containerRect = container.getBoundingClientRect();

      tooltipElement.style.left = (rect.left - containerRect.left +
    rect.width + 10) + 'px';
      tooltipElement.style.top = (rect.top - containerRect.top - 10) +
    'px';
}
```

Sensores Interactivos con Animaciones

```
javascript
    sensorData.forEach((sensor, index) => {
      const position = getSensorPosition(index, sensorData.length);
      const status = getSensorStatus(sensor);
      const color = getStatusColor(status);
```

```
        const sensorCircle = el("circle", {
          class: "sensor-point",
          cx: position.x, cy: position.y, r: "8",
          fill: color, stroke: "#fff", "stroke-width": "2",
          style: `
            cursor: pointer;
            transition: all 0.3s ease;
            filter: drop-shadow(0 2px 4px rgba(0,0,0,0.3));
            ${status === 'alert' ? 'animation: pulse 2s infinite;' : ''}
          `,
          "data-sensor": sensor.tipo_sensor,
          "data-value": sensor.valor,
          title: `${sensor.tipo_sensor}: ${sensor.valor}${sensor.unidad ||
      ''}`
        });
});
```

Sistema de Estilos CSS

```
css
      @keyframes pulse {
        0% { transform: scale(1); opacity: 1; }
        50% { transform: scale(1.1); opacity: 0.7; }
```

```
      100% { transform: scale(1); opacity: 1; }
    }

    .sensor-point:hover {
      r: 12; /* Aumenta tamaño al hover */
      filter: drop-shadow(0 4px 8px rgba(0,0,0,0.4)) !important;
}
```