Manual del Desarrollador
Huerto Urbano Inteligente IoT

Azuero Maldonado Ronald Alejandro
Toro Ramon Diego Steeven

9 de junio de 2025

Índice

1. Introducción

Este manual está dirigido a desarrolladores y técnicos que deseen comprender, mantener, modificar o extender el sistema de Huerto Urbano Inteligente basado en Node-RED. Aquí se detallan la arquitectura del flujo, la lógica de los nodos, la gestión de datos y las consideraciones para el despliegue.

2. Arquitectura del Flujo en Node-RED

El proyecto de Huerto Urbano Inteligente en Node-RED se organiza en un único flujo ('tab') denominado "Dashboard Definitivo v5.1". Este flujo integra la simulación de sensores, la lógica de control, la adquisición de datos externos (clima), y la presentación en un dashboard interactivo.

La comunicación de datos entre los diferentes componentes del flujo se realiza principalmente a través de `flow context variables` y `msg.payload`.

2.1. Nodos Principales y su Función

A continuación, se describen los nodos clave del flujo y su propósito.

1. **Librerías Externas (CSS/JS)** (`ui_template`, ID: `4e91dec7a7e66d5b`)
   - **Propósito:** Carga hojas de estilo (CSS) y scripts JavaScript globales necesarios para la apariencia y funcionalidad del dashboard.
   - **Detalles:** Incluye Bootstrap para el diseño responsivo, SweetAlert2 para pop-ups, Toastr para notificaciones "toast", Font Awesome para iconos, y ECharts para la visualización de gráficos.
   - **Configuración Clave:** Inicializa las opciones de Toastr para notificaciones consistentes en todo el dashboard.
   - **Código Relevante:**

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Manrope:wght@400
    ;500;700&display=swap" rel="stylesheet">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/
    bootstrap.min.css" rel="stylesheet">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/sweetalert2@11/
    dist/sweetalert2.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    awesome/4.7.0/css/font-awesome.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr
    .js/2.1.4/toastr.min.css">

<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

```
14
15 <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
16
17 <script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/2.1.4/toastr
       .min.js"></script>
18
19 <script src="https://cdn.jsdelivr.net/npm/echarts@5.5.0/dist/echarts.min.
       js"></script>
20
21 <script>
22     // Configuramos Toastr una sola vez para todo el dashboard
23     // Usamos un pequeo setTimeout para asegurar que Toastr est
           completamente cargado y disponible
24     setTimeout(function() {
25         if (typeof toastr !== 'undefined') { // Add a check to be extra
               safe
26             toastr.options = {
27                 "closeButton": true,
28                 "debug": false,
29                 "newestOnTop": true,
30                 "progressBar": true,
31                 "positionClass": "toast-top-right",
32                 "preventDuplicates": false,
33                 "onclick": null,
34                 "showDuration": "300",
35                 "hideDuration": "1000",
36                 "timeOut": "4000",
37                 "extendedTimeOut": "1000",
38                 "showEasing": "swing",
39                 "hideEasing": "linear",
40                 "showMethod": "fadeIn",
41                 "hideMethod": "fadeOut"
42             }
43             console.log("Toastr␣configured␣successfully␣with␣timeout.");
44         } else {
45             console.error("Toastr␣is␣still␣not␣defined␣after␣timeout.␣
                   Manual␣inspection␣needed.");
46         }
47     }, 100); // 100ms delay for configuration
48 </script>
```

Listing 1: Fragmento de código del nodo 'Librerías Externas (CSS/JS)'

2. INICIALIZADOR (`function`, ID: `b697b3dcd1a877a8`)

- **Propósito:** Configura y reinicia todas las variables de contexto de 'flow' al desplegar el flujo. Esto asegura que el sistema comience con un estado conocido y consistente.
- **Variables Inicializadas:**
  - `tick`: Contador para la simulación.

- `tankLevel`, `soilMoisture`, `nutriente_A`, `nutriente_B`, `soil_nutrient_level`, `plantHealth`: Estados simulados de sensores y salud.
- `pumpOn`, `lightsOn`: Estado de los actuadores.
- `soil_sensor_status`: Estado del sensor de humedad ('ok' o 'error').
- `config_umbral_riego`, `config_pump_override`, `config_lights_override`, `config_lights_intensity`: Variables de configuración para el control.
- `temperaturaData`, `humedadData`, `timestampData`: Arrays para datos de gráficos.
- `event_log`: Array para el registro de eventos.
- `real_weather`: Objeto para datos del clima real.

- **Activación:** Se dispara una vez al inicio del flujo por un nodo `inject` (."Al Desplegar").
- **Código Relevante:**

```
// Inicializar todas las variables de flow al desplegar el flujo
// Aseguramos que TODAS las variables usadas en el Agregador de Datos tengan un valor inicial
flow.set('tick', 0);
flow.set('tankLevel', 80); // Nivel de tanque inicial
flow.set('soilMoisture', 65); // Humedad del suelo inicial
flow.set('pumpOn', false); // Estado inicial de la bomba
flow.set('lightsOn', false); // Estado inicial de las luces
flow.set('nutriente_A', 75); // Nivel inicial de nutriente A
flow.set('nutriente_B', 60); // Nivel inicial de nutriente B
flow.set('soil_nutrient_level', 90); // Nivel inicial de nutrientes en el suelo
flow.set('soil_sensor_status', 'ok'); // Estado inicial del sensor de suelo: 'ok', 'error'
flow.set('plantHealth', 100); // Salud inicial de la planta (100%)

// Variables de configuracin iniciales
flow.set('config_umbral_riego', 35); // Umbral de riego por defecto
flow.set('config_pump_override', 'auto'); // Modo de bomba por defecto
flow.set('config_lights_override', 'auto'); // Modo de luces por defecto
flow.set('config_lights_intensity', 100); // Intensidad de luces por defecto

// Datos para las grficas (arrays vacos iniciales)
flow.set('temperaturaData', []);
flow.set('humedadData', []);
flow.set('timestampData', []);

// Registro de eventos (array vaco inicial)
flow.set('event_log', []);

// Datos para el clima real (objeto vaco inicial con defaults)
flow.set('real_weather', {
    label: "Clima Exterior",
    description: "Cargando datos...",
    iconUrl: "",
```

```
    temp: "- C",
    humidity: "- %"
});

node.warn("INICIALIZADOR: Todas las variables de flow inicializadas.");

return null;
```

Listing 2: Fragmento de código del nodo 'INICIALIZADOR'

3. **Disparador de Simulación (5s)** (`inject`, ID: `e41a00cc6c48e67e`)

   - **Propósito:** Genera un mensaje cada 1 segundo (según la configuración del JSON, aunque el nombre del nodo indica 5s) para activar la lógica de control y simulación de sensores.
   - **Configuración:** Repetir cada 1 segundo ('repeat": "1«).
   - **Lógica de Control** (`function`, ID: `f25357e13f5b1a86`)
     - **Propósito:** Implementa la lógica de control para la bomba de riego y las luces de crecimiento, basándose en los datos simulados y las configuraciones del usuario.
     - **Control de Bomba:**
       - Si `pumpOverride` es 'on' o 'off', fuerza el estado.
       - Si es 'auto', activa la bomba si `soilMoisture` es menor que `config_umbral_riego`, y la desactiva si `soilMoisture` supera el 80 %.
     - **Control de Luces:**
       - Si `lightsOverride` es 'on' o 'off', fuerza el estado.
       - Si es 'auto', activa las luces entre las 6 PM y las 6 AM (basado en la hora actual del sistema).
     - **Salida:** Envía mensajes al Logger de Eventos si el estado de la bomba o las luces cambia.
     - **Código Relevante:**

```
const umbralRiego = flow.get('config_umbral_riego') || 35;
const pumpOverride = flow.get('config_pump_override') || 'auto';
let soilMoisture = flow.get('soilMoisture') || 65;
let oldPumpStatus = flow.get('pumpOn') || false;
let newPumpStatus = false;
if (pumpOverride === 'on') { newPumpStatus = true; }
else if (pumpOverride === 'off') { newPumpStatus = false; }
else {
    if (soilMoisture < umbralRiego) { newPumpStatus = true; }
    else if (soilMoisture > 80) { newPumpStatus = false; }
    else { newPumpStatus = oldPumpStatus; }
}
if (newPumpStatus === true && oldPumpStatus === false) { node.send([null,
    {payload: "Bomba de riego activada."}]); }
```

```
else if (newPumpStatus === false && oldPumpStatus === true) { node.send([
    null, {payload: "Bomba de riego detenida."}]); }

const lightsOverride = flow.get('config_lights_override') || 'auto';
let oldLightsStatus = flow.get('lightsOn') || false;
let newLightsStatus = false;
if (lightsOverride === 'on') { newLightsStatus = true; }
else if (lightsOverride === 'off') { newLightsStatus = false; }
else {
    const hour = new Date().getHours();
    newLightsStatus = (hour >= 18 || hour < 6);
}
if (newLightsStatus === true && oldLightsStatus === false) { node.send([
    null, {payload: "Luces de crecimiento encendidas."}]); }
else if (newLightsStatus === false && oldLightsStatus === true) { node.
    send([null, {payload: "Luces de crecimiento apagadas."}]); }

flow.set('pumpOn', newPumpStatus);
flow.set('lightsOn', newLightsStatus);

return [msg, null];
```

Listing 3: Fragmento de código del nodo 'Lógica de Control'

- **Logger de Eventos** (`function`, ID: `25d2ed934e93fbf9`)
  - **Propósito:** Almacena un registro de eventos importantes del sistema en la variable de 'flow' `event_log`.
  - **Funcionalidad:** Limita el tamaño del log a un máximo de 20 entradas, eliminando las más antiguas.
  - **Código Relevante:**

```
const MAX_LOG_SIZE = 20;

let log = flow.get('event_log') || [];

const timestamp = new Date().toLocaleTimeString('es-EC');
const newEntry = {
    time: timestamp,
    message: msg.payload
};

log.unshift(newEntry);

if (log.length > MAX_LOG_SIZE) {
    log.pop();
}

flow.set('event_log', log);

return null;
```

Listing 4: Fragmento de código del nodo 'Logger de Eventos'

- **Simulador de Sensores** (`function`, ID: `48a1e2359a5d584b`)
  - **Propósito:** Simula las lecturas de los sensores del huerto (temperatura, humedad ambiente, humedad del suelo, nivel del tanque, nutrientes, salud de la planta) y actualiza las variables de 'flow'.
  - **Lógica de Simulación:**
    - **Temp/Humedad Ambiente:** Simula un ciclo día/noche usando funciones seno/coseno y un contador 'tick'.
    - **Humedad del Suelo:** Aumenta si la bomba está activa, disminuye lentamente si está inactiva. Se congela si `soil_sensor_status` es 'error'.
    - **Nivel del Tanque:** Disminuye si la bomba está activa.
    - **Nutrientes:** Disminuyen lentamente con el tiempo (Nutriente A, Nutriente B, y Nutrientes Generales del Suelo).
    - **Salud de la Planta:** Calcula un cambio basado en la proximidad de la humedad del suelo, nutrientes del suelo y temperatura a sus rangos óptimos. Penaliza fuertemente fuera de rango y recompensa significativamente dentro de rango.
  - **Actualización de Datos de Gráficos:** Almacena los últimos 40 puntos de datos de temperatura y humedad para la visualización en el dashboard.
  - **Depuración:** Incluye mensajes `node.warn` para monitorear el estado de las variables simuladas en la consola de depuración de Node-RED.
  - **Código Relevante:**

```
// Obtener los estados actuales de las variables de flow
let pumpOn = flow.get('pumpOn') || false;
let tankLevel = flow.get('tankLevel') || 80;
let soilMoisture = flow.get('soilMoisture') || 65;
let tick = flow.get('tick') || 0;
let nutrienteA = flow.get('nutriente_A') || 75;
let nutrienteB = flow.get('nutriente_B') || 60;
let soilNutrients = flow.get('soil_nutrient_level') || 90;
let soilSensorStatus = flow.get('soil_sensor_status') || 'ok';
let plantHealth = flow.get('plantHealth') || 100; // Obtener el valor
    actual de la salud de la planta

// --- Simulacin de Temperatura y Humedad Ambiente ---
const tempBase = 20;
const tempAmplitude = 8;
const temp = tempBase + Math.sin(tick * 0.1) * tempAmplitude; // Simulacin
     de ciclo da/noche
const humidityBase = 60;
const humidityAmplitude = 20;
const humidity = humidityBase - Math.sin(tick * 0.1) * humidityAmplitude;
    // Simulacin de ciclo da/noche
```

```
19
20 // --- Lgica PRINCIPAL de Humedad del Suelo y Tanque ---
21 if (soilSensorStatus === 'ok') {
22     if (pumpOn) {
23         soilMoisture += 5; // Aumenta la humedad del suelo cuando la bomba
               est encendida
24         tankLevel -= 2; // El tanque disminuye cuando la bomba est activa
25     } else {
26         soilMoisture -= 0.5; // Disminuye lentamente si la bomba est
               apagada
27     }
28 } else {
29     // Si hay un error en el sensor de suelo, el valor se "congela"
30     // (no se actualiza, se mantiene el ltimo valor de flow.get('
           soilMoisture'))
31 }
32
33 // --- Consumo de Nutrientes A y B ---
34 nutrienteA -= 0.2; // Consumo continuo
35 nutrienteB -= 0.3; // Consumo continuo
36 soilNutrients -= 0.1; // Consumo lento de nutrientes en el suelo
37
38 // --- Lgica de la Salud de la Planta (AJUSTADA para EXTREMA ROBUSTEZ y
       RECUPERACIN) ---
39 let healthChange = 0; // Se inicializa a 0 en cada tick
40
41 // Impacto de la humedad del suelo
42 if (soilMoisture < 60) { // Demasiado seco
43     healthChange -= 3.15; // Penalizacin MNIMA por sequedad
44 } else if (soilMoisture > 95) { // Demasiado hmedo
45     healthChange -= 4.20; // Penalizacin MNIMA por exceso de agua
46 } else { // Rango ptimo de humedad (30-85)
47     healthChange += 5.0; // ENORME recuperacin si la humedad es ptima!
48 }
49
50 // Impacto de los nutrientes del suelo
51 if (soilNutrients < 40) { // Nutrientes bajos
52     healthChange -= 3.05; // Penalizacin MNIMA por nutrientes bajos
53 } else if (soilNutrients > 95) { // Exceso de nutrientes
54     healthChange -= 3.03; // Penalizacin MNIMA por exceso de nutrientes
55 } else { // Rango ptimo de nutrientes (20-95)
56     healthChange += 3.5; // ENORME recuperacin si los nutrientes son ptimos
           !
57 }
58
59 // Impacto de la temperatura
60 if (temp < 15 || temp > 28) { // Temperatura fuera de rango ptimo
61     healthChange -= 1.02; // Penalizacin MNIMA por temperatura extrema
62 } else { // Temperatura ptima
63     healthChange += 3.0; // ENORME recuperacin si la temperatura es ptima!
64 }
```

```
65
66 plantHealth += healthChange; // Aplica el cambio a la salud de la planta
67
68 // Limitar todos los valores para que no sean negativos o excedan el 100%
69 if (soilMoisture > 100) soilMoisture = 100;
70 if (soilMoisture < 0) soilMoisture = 0;
71 if (tankLevel < 0) tankLevel = 0;
72 if (nutrienteA < 0) nutrienteA = 0;
73 if (nutrienteB < 0) nutrienteB = 0;
74 if (soilNutrients < 0) soilNutrients = 0;
75 if (plantHealth > 100) plantHealth = 100; // Limitar salud a 100%
76 if (plantHealth < 0) plantHealth = 0; // Limitar salud a 0%
77
78 // Guardar los estados actualizados en las variables de flow
79 flow.set('soilMoisture', soilMoisture);
80 flow.set('tankLevel', tankLevel);
81 flow.set('tick', tick + 1);
82 flow.set('nutriente_A', nutrienteA);
83 flow.set('nutriente_B', nutrienteB);
84 flow.set('soil_nutrient_level', soilNutrients);
85 flow.set('plantHealth', plantHealth); // Guardar la salud de la planta
       actualizada
86
87 // Preparar datos individuales para el agregador (usando parseFloat para
       asegurar formato)
88 flow.set('sim_temp', parseFloat(temp.toFixed(1)));
89 flow.set('sim_hum', parseFloat(humidity.toFixed(1)));
90 flow.set('sim_soil', parseFloat(soilMoisture.toFixed(0)));
91 flow.set('sim_tank', parseFloat(tankLevel.toFixed(0)));
92 flow.set('sim_n_a', parseFloat(nutrienteA.toFixed(1)));
93 flow.set('sim_n_b', parseFloat(nutrienteB.toFixed(1)));
94 flow.set('sim_soil_nutrients', parseFloat(soilNutrients.toFixed(1)));
95 flow.set('sim_soil_status', soilSensorStatus);
96 flow.set('sim_plant_health', parseFloat(plantHealth.toFixed(1))); // Para
       el agregador
97
98 // Actualizar datos para las grficas
99 const MAX_DATAPOINTS = 40;
100 let tempData = flow.get('temperaturaData') || [];
101 let humData = flow.get('humedadData') || [];
102 let xAxisData = flow.get('timestampData') || [];
103 let timestamp = new Date().toLocaleTimeString('es-EC', { hour: '2-digit',
       minute: '2-digit' });
104
105 tempData.push(parseFloat(temp.toFixed(1)));
106 humData.push(parseFloat(humidity.toFixed(1)));
107 xAxisData.push(timestamp);
108
109 if (tempData.length > MAX_DATAPOINTS) {
110     tempData.shift();
111     humData.shift();
```

```
    xAxisData.shift();
}

flow.set('temperaturaData', tempData);
flow.set('humedadData', humData);
flow.set('timestampData', xAxisData);

// --- Consola de Depuracin para Simulador de Sensores (se ver en Node-RED Debug Sidebar) ---
node.warn("--- DEBUG SIMULADOR DE SENSORES ---");
node.warn("Cambio en Salud (healthChange): " + healthChange.toFixed(2));
node.warn("Salud de la Planta (calculada): " + plantHealth.toFixed(1));
node.warn("Humedad Suelo actual: " + soilMoisture.toFixed(0));
node.warn("Nutrientes Suelo actual: " + soilNutrients.toFixed(1));
node.warn("Temperatura actual: " + temp.toFixed(1));
node.warn("-------------------------------------");

return msg;
```

Listing 5: Fragmento de código del nodo 'Simulador de Sensores'

- **Disparador de Clima (15m)** (`inject`, ID: `abaf5ed597e2e7da`)
  - **Propósito:** Obtiene datos del clima exterior cada 15 minutos.
  - **Configuración:** Repetir cada 900 segundos ('repeat": "900«).
  - **Clima en Pinas, EC** (`openweathermap`, ID: `e9674d7f4b69cc0e`)
    - **Propósito:** Nodo para obtener datos climáticos de OpenWeatherMap.
    - **Configuración:** Se especifica la latitud ('-3.6833') y longitud ('-79.6833') de Piñas, Ecuador.
    - **Nota:** Requiere una clave API de OpenWeatherMap configurada en las propiedades del nodo. Esta clave no se muestra en el JSON.
  - **Guarda Datos del Clima** (`function`, ID: `036bcb8e5a377061`)
    - **Propósito:** Procesa la respuesta del nodo OpenWeatherMap y la formatea en un objeto 'real$_w$*eather*'*queseguardaenelcontextode*'*flow*'.**Campos Guar**

- **Código Relevante:**

```
const city = msg.location.city;
const description = msg.payload.detail;
const icon = msg.payload.icon;
const temp = msg.payload.tempc;
const humidity = msg.payload.humidity;

const weatherData = {
    label: "Clima Exterior en " + city,
    description: description.charAt(0).toUpperCase() + description.slice(1),
    iconUrl: "https://openweathermap.org/img/w/" + icon + ".png",
```

```
    temp: temp + " C",
    humidity: humidity + " %"
};

flow.set('real_weather', weatherData);

return msg;
```

Listing 6: Fragmento de código del nodo 'Guarda Datos del Clima'

- **Agregador de Datos** (`function`, ID: `604e3cd5c0a162de`)
  - **Propósito:** Recopila todas las variables de 'flow' relevantes (tanto simuladas como configuraciones y datos externos) y las consolida en un único objeto 'msg.payload'. Este payload es el que se envía al nodo 'ui$_t$*emplate*'*deldashboard*.**Validación de Datos:***Aseguraquetodaslasvariablesten*

- **Cálculo de Alertas:** Determina las alertas booleanas para el nivel del tanque, humedad del suelo (basado en el umbral configurado) y nutrientes bajos.
- **Estructura del Payload:** El payload final tiene una estructura jerárquica ('invernadero', 'sistema', 'climaReal', 'config', 'log') para facilitar el acceso desde el frontend.
- **Depuración:** Incluye mensajes `node.warn` para mostrar el payload completo saliente, útil para depuración.
- **Código Relevante:**

```
// Obtener todos los valores de flow, usando valores predeterminados
    seguros
const tankLevel = flow.get('sim_tank') || 80; // Default to 80 if
    undefined
const soilMoisture = flow.get('sim_soil') || 65; // Default to 65 if
    undefined
const umbralRiego = flow.get('config_umbral_riego') || 35; // Default to
    35 if undefined
const soilNutrients = flow.get('soil_nutrient_level') || 90; // Default to
     90 if undefined
const plantHealth = flow.get('plantHealth') || 100; // Default to 100 if
    undefined
const soilSensorStatus = flow.get('sim_soil_status') || 'ok'; // Default
    to 'ok' if undefined
const simTemp = flow.get('sim_temp') || 25; // Default to 25 if undefined
const simHum = flow.get('sim_hum') || 60; // Default to 60 if undefined
const pumpOn = flow.get('pumpOn') || false; // Default to false if
    undefined
const lightsOn = flow.get('lightsOn') || false; // Default to false if
    undefined
const lightsIntensity = flow.get('config_lights_intensity') || 100; //
    Default to 100 if undefined
const pumpOverride = flow.get('config_pump_override') || 'auto'; //
    Default to 'auto' if undefined
const lightsOverride = flow.get('config_lights_override') || 'auto'; //
    Default to 'auto' if undefined
```

```
15 const eventLog = flow.get('event_log') || []; // Default to empty array if
       undefined
16
17 // Clculos de alerta (ahora usan los valores por defecto si los originales
       eran undefined)
18 const tankAlert = tankLevel < 20;
19 const soilMoistureAlert = soilMoisture < umbralRiego; // Usa el umbral
      real o default
20 const nutrientsLowAlert = soilNutrients < 20; // Umbral para nutrientes
      bajos (ej. < 20%)
21
22 // Obtener datos de la grfica, asegurndose de que sean arrays
23 const temperaturaData = flow.get('temperaturaData') || [];
24 const humedadData = flow.get('humedadData') || [];
25 const timestampData = flow.get('timestampData') || [];
26
27 // Obtener datos del clima real, asegurndose de que sea un objeto
28 const realWeather = flow.get('real_weather') || {
29     label: "Cargando Clima...",
30     description: "N/A",
31     iconUrl: "",
32     temp: "-- C",
33     humidity: "-- %"
34 };
35
36 // --- Depuracin detallada de variables de entrada para Agregador de Datos
       ---
37 node.warn("--- DEBUG AGREGADOR DE DATOS (Variables de Entrada) ---");
38 node.warn("sim_temp: " + simTemp);
39 node.warn("sim_hum: " + simHum);
40 node.warn("soilMoisture: " + soilMoisture);
41 node.warn("soilNutrients: " + soilNutrients);
42 node.warn("plantHealth: " + plantHealth);
43 node.warn("soilSensorStatus: " + soilSensorStatus);
44 node.warn("tankLevel: " + tankLevel);
45 node.warn("pumpOn: " + pumpOn);
46 node.warn("lightsOn: " + lightsOn);
47 node.warn("umbralRiego (config): " + umbralRiego);
48 node.warn("pumpOverride (config): " + pumpOverride);
49 node.warn("lightsOverride (config): " + lightsOverride);
50 node.warn("lightsIntensity (config): " + lightsIntensity);
51 node.warn("---------------------------------------");
52
53
54 const payload = {
55     invernadero: {
56         temperatura: simTemp,
57         humedadAire: simHum,
58         humedadSuelo: soilMoisture,
59         nutrientesSuelo: soilNutrients,
60         humedadSueloStatus: soilSensorStatus,
```

```
        nivelTanque: tankLevel,
        tankAlert: tankAlert,
        soilMoistureAlert: soilMoistureAlert,
        nutrientsLowAlert: nutrientsLowAlert,
        plantHealth: plantHealth,
        // Asegurar que nutrienteA y nutrienteB tambin tengan fallbacks
        nutrienteA: flow.get('nutriente_A') || 75,
        nutrienteB: flow.get('nutriente_B') || 60,
        chartData: {
            seriesData: {
                temperatura: temperaturaData,
                humedad: humedadData
            },
            xAxisData: timestampData
        }
    },
    sistema: {
        bombaActiva: pumpOn,
        lucesActivas: lightsOn
    },
    climaReal: realWeather,
    config: {
        umbralRiego: umbralRiego,
        pumpOverride: pumpOverride,
        lightsOverride: lightsOverride,
        lightsIntensity: lightsIntensity
    },
    log: eventLog
};

msg.payload = payload;

// --- Imprimir el payload completo en el Debug Sidebar para depuracin ---
node.warn("--- DEBUG AGREGADOR DE DATOS (Payload Saliente Completo) ---");
node.warn(JSON.stringify(payload, null, 2)); // Imprime el objeto payload
    formateado
node.warn("-----------------------------------------");


return msg;
```

Listing 7: Fragmento de código del nodo 'Agregador de Datos'

- **Distribuidor de Controles** (`switch`, ID: `c52ce99c2e332e82`)
  - **Propósito:** Dirige los mensajes entrantes desde los controles del dashboard a los nodos `change` o `function` correspondientes para actualizar las variables de configuración.
  - **Reglas:** Basado en el `msg.topic` (ej., 'umbral_riego', 'pump_override', 'lights_intensity', 'abonar_tierra', 'simular_fallo').
- **Nodos de Guardado de Configuración** (`change`, IDs: `8149f5e4b6abdaa3`,

7329a45d44e9a384, 3db9d8ed0740fef0, 4127cca90e950eb1)

- **Propósito:** Estos nodos `change` son utilizados para tomar el `msg.payload` recibido de un control del dashboard y guardar su valor directamente en la variable de `flow` correspondiente (ej., `config_umbral_riego`, `config_pump_override`, etc.).

- **Abonar Tierra** (`function`, ID: `b1c2d3e4f5a6b7c8`)
  - **Propósito:** Simula la adición de nutrientes a la tierra, incrementando el valor de `nutriente_A`, `nutriente_B` y `soil_nutrient_level` en las variables de `flow`. También genera un mensaje para el Logger de Eventos.
  - **Código Relevante:**

```
let nutrienteA = flow.get('nutriente_A') || 0;
let nutrienteB = flow.get('nutriente_B') || 0;
let soilNutrients = flow.get('soil_nutrient_level') || 0;

nutrienteA = Math.min(100, nutrienteA + 20); // Simula adicin de nutriente
    A
nutrienteB = Math.min(100, nutrienteB + 15); // Simula adicin de nutriente
    B
soilNutrients = Math.min(100, soilNutrients + 25); // Simula adicin a
   nutrientes del suelo

flow.set('nutriente_A', nutrienteA);
flow.set('nutriente_B', nutrienteB);
flow.set('soil_nutrient_level', soilNutrients);

msg.payload = "Tierra abonada. Nutrientes restablecidos.";
return [null, msg]; // Mensaje para el logger.
```

Listing 8: Fragmento de código del nodo 'Abonar Tierra'

- **Alternar Fallo de Sensor** (`function`, ID: `c1d2e3f4a5b6c7d9`)
  - **Propósito:** Permite simular un fallo en el sensor de humedad del suelo, alternando el estado de la variable `soil_sensor_status` entre 'ok' y 'error'. Genera un mensaje para el Logger de Eventos.
  - **Impacto:** Cuando está en 'error', el nodo "Simulador de Sensores"detiene la actualización de la humedad del suelo.
  - **Código Relevante:**

```
let currentStatus = flow.get('soil_sensor_status') || 'ok';

if (currentStatus === 'ok') {
    flow.set('soil_sensor_status', 'error');
    msg.payload = "Fallo simulado en sensor de humedad.";
} else {
    flow.set('soil_sensor_status', 'ok');
    msg.payload = "Sensor de humedad recuperado.";
```

```
}

return msg;
```

Listing 9: Fragmento de código del nodo 'Alternar Fallo de Sensor'

- **Single Page Dashboard** (`ui_template`, ID: `fad068f04fb6ab06`)
  - **Propósito:** Este es el corazón de la interfaz de usuario. Renderiza el HTML, CSS y JavaScript para el dashboard completo.
  - **Funcionamiento:** Recibe el payload de datos del .Agregador de Datosz usa JavaScript (enlazado a la sintaxis del dashboard de Node-RED) para actualizar dinámicamente todos los elementos visuales (medidores, gráficos, tarjetas de información, logs).
  - **CSS:** Incluye estilos personalizados para una apariencia moderna y responsiva.
  - **JavaScript (en el formato):**
    - Manejo de actualizaciones de datos: 'scope.$watch('msg', function(msg)...)'.Lóg
- ◇ Configuración y renderizado de gráficos ECharts.
- ◇ Actualización del estado de la bomba y las luces.
- ◇ Renderizado del registro de eventos.
- ◇ Manejo de interacciones del usuario (botones, sliders) para enviar mensajes de vuelta al flujo de Node-RED (utilizando 'scope.send(topic: ..., payload: ...)').
- ◇ **Notificaciones Toastr y SweetAlert2:** Implementación de notificaciones pop-up para una mejor experiencia de usuario.

  - **Código Relevante (Fragmento inicial del HTML y CSS):**

```
<script src="https://cdn.jsdelivr.net/npm/echarts@5.5.0/dist/echarts.min.
    js"></script>

<div class="hydro-tech-dashboard">
    <header class="dashboard-header">
        <h1>Dashboard Huerto Urbano Inteligente IOT</h1>
        <div id="current-time"></div>
    </header>
    <aside class="sidebar">
        <div class="grid-item">
            <div class="info-card" id="pump-card"></div>
        </div>
        <div class="grid-item">
            <div class="info-card" id="lights-card"></div>
        </div>
        <div class="grid-item␣weather-item">
            <div class="weather-card" id="weather-card"></div>
        </div>
        <div class="grid-item␣event-log-item" id="event-log-card"></div>
    </aside>
    <main class="main-content">
        <div class="grid-item␣chart-container" id="chart-container"></div>
```

```
        <div class="grid-item" id="soil-gauge-card"></div>
        <div class="grid-item" id="tank-gauge-card"></div>
        <div class="grid-item" id="soil-nutrient-card"></div>
        <div class="grid-item" id="plant-health-card"></div>
        <div class="grid-item config-item" id="config-card"></div>
    </main>
</div>

<style>
    :root {
        --bg-main: #f4f7f9;
        --bg-card: #ffffff;
        --border-color: #dee2e6;
        --text-primary: #2c3e50;
        --text-secondary: #7f8c8d;
        --accent-green: #27ae60;
        --accent-blue: #2980b9;
        --accent-red: #c0392b;
        --accent-yellow: #f39c12;
        --soil-color: #967259;
        --nutrient-color: #8e44ad;
        --health-color: #2ecc71;
    }

    .hydro-tech-dashboard {
        font-family: 'Manrope', sans-serif;
        display: grid;
        grid-template-columns: 320px 1fr;
        grid-template-rows: auto 1fr;
        grid-template-areas: "header header""sidebar main";
        gap: 15px;
        width: 100%;
        height: 100%;
        padding: 15px;
        box-sizing: border-box;
        background-color: var(--bg-main);
        color: var(--text-primary);
    }

    .grid-item {
        background-color: var(--bg-card);
        color: var(--text-primary);
        border-radius: 8px;
        border: 1px solid var(--border-color);
        padding: 20px;
        box-sizing: border-box;
        transition: all 0.3s ease;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: stretch;
```

```
73          box-shadow: 0 4px 12px rgba(0, 0, 0, 0.08);
74          position: relative;
75      }
76
77      .dashboard-header {
78          grid-area: header;
79          display: flex;
80          justify-content: space-between;
81          align-items: center;
82          padding: 0 10px;
83          margin-bottom: 5px;
84      }
85
86      .dashboard-header h1 {
87          font-size: 1.8em;
88          color: var(--text-primary);
89          margin: 0;
90      }
91
92      #current-time {
93          font-size: 1.2em;
94          color: var(--text-secondary);
95      }
96
97      .sidebar {
98          grid-area: sidebar;
99          display: grid;
100         grid-template-rows: auto auto 1fr auto;
101         gap: 15px;
102     }
103
104     .main-content {
105         grid-area: main;
106         display: grid;
107         grid-template-columns: repeat(3, 1fr);
108         grid-auto-rows: minmax(180px, auto);
109         gap: 15px;
110     }
111
112     .chart-container {
113         grid-column: 1 / 4;
114         grid-row: span 2;
115         min-height: 400px;
116     }
117
118     .config-item {
119         grid-column: 1 / 4;
120     }
121
122     #soil-gauge-card,
123     #tank-gauge-card,
```

```
124     #soil-nutrient-card,
125     #plant-health-card {
126         grid-column: span 1;
127     }
128 
129     .event-log-item {
130         min-height: 200px;
131     }
132 
133     .info-card,
134     .weather-card,
135     .config-container,
136     .tank-indicator,
137     .event-log-container {
138         height: 100%;
139         width: 100%;
140         display: flex;
141         flex-direction: column;
142         justify-content: center;
143         align-items: stretch;
144         text-align: center;
145     }
146 
147     .info-card .label,
148     .config-container .title,
149     .tank-indicator .title,
150     .event-log-container .title {
151         font-size: 0.9em;
152         color: var(--text-secondary);
153         text-transform: uppercase;
154         margin-bottom: 10px;
155         text-align: center;
156         flex-shrink: 0;
157         font-weight: 600;
158     }
159 
160     .info-card .value {
161         font-size: 1.6em;
162         font-weight: 700;
163         display: flex;
164         align-items: center;
165         justify-content: center;
166     }
167 
168     .weather-card {
169         justify-content: space-around;
170         background: linear-gradient(135deg, var(--accent-blue) 0%, #6dd5ed
                100%);
171         /* Fondo degradado */
172         color: white;
173         /* Texto blanco para contraste */
```

```
        text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.2);
        padding: 15px;
        /* Ajustar padding */
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
    }

    .weather-card .location-label {
        /* Nuevo estilo para la ubicacin */
        font-size: 1.1em;
        font-weight: 500;
        margin-bottom: 10px;
        opacity: 0.9;
    }

    .weather-card .weather-main-info {
        /* Contenedor principal de icono y temperatura */
        display: flex;
        align-items: center;
        margin-bottom: 10px;
    }

    .weather-icon img {
        width: 90px;
        /* Icono ms grande */
        filter: drop-shadow(2px 2px 5px rgba(0, 0, 0, 0.3));
        /* Sombra para resaltar */
        animation: float 3s ease-in-out infinite;
        /* Animacin flotante */
        margin-right: 15px;
        /* Espacio a la derecha del icono */
    }
```

Listing 10: Fragmento de código HTML y CSS del nodo 'Single Page Dashboard'

El código JavaScript completo de este nodo es muy extenso y no se incluye aquí para brevedad, pero se encuentra en el flujo JSON proporcionado.

3. Variables de Contexto de Flow

Las variables de contexto de `flow` son fundamentales para mantener el estado del sistema y permitir que los diferentes nodos compartan información.

- `tick`: Contador interno para la simulación de tiempo.
- `tankLevel`: Nivel de agua en el tanque (0-100 %).
- `soilMoisture`: Porcentaje de humedad en el suelo (0-100 %).

- `pumpOn`: Booleano, indica si la bomba de riego está activa.
- `lightsOn`: Booleano, indica si las luces de crecimiento están encendidas.
- `nutriente_A`, `nutriente_B`: Niveles de nutrientes A y B (simulados).
- `soil_nutrient_level`: Nivel general de nutrientes en el suelo (0-100 %).
- `soil_sensor_status`: Estado del sensor de humedad ('ok' o 'error').
- `plantHealth`: Porcentaje de salud de la planta (0-100 %).
- `config_umbral_riego`: Umbral de humedad para el riego automático.
- `config_pump_override`: Modo de control de la bomba ('auto', 'on', 'off').
- `config_lights_override`: Modo de control de las luces ('auto', 'on', 'off').
- `config_lights_intensity`: Intensidad de las luces (0-100 %).
- `temperaturaData`, `humedadData`, `timestampData`: Arrays que almacenan datos históricos para los gráficos.
- `event_log`: Array de objetos, cada uno con `time` y `message` de los eventos del sistema.
- `real_weather`: Objeto que contiene los datos del clima exterior.
- `sim_temp`, `sim_hum`, `sim_soil`, `sim_tank`, `sim_n_a`, `sim_n_b`, `sim_soil_nutrients`, `sim_soil_status`, `sim_plant_health`: Variables temporales usadas por el Simulador de Sensores antes de ser agregadas al payload final.

4. Dependencias y Configuraciones Externas

4.1. Nodos Node-RED Requeridos

Asegúrese de tener instalados los siguientes nodos en su instancia de Node-RED:

- `node-red-dashboard` (para los nodos `ui_template` y otros componentes del dashboard).
- `node-red-node-openweathermap` (para el nodo OpenWeatherMap).

Puede instalarlos a través de la paleta de Node-RED (Manage Palette) o desde la línea de comandos en su directorio de Node-RED:

```
npm install node-red-dashboard
npm install node-red-node-openweathermap
```

4.2. Librerías Frontend (CSS/JS)

Las librerías se cargan directamente desde CDN en el nodo "Librerías Externas (CSS/JS)". Estas incluyen:

- Google Fonts (Manrope)
- Bootstrap (v5.3.3)
- SweetAlert2 (v11)
- Font Awesome (v4.7.0)
- Toastr (v2.1.4)
- jQuery (v3.7.1)
- ECharts (v5.5.0)

No se requiere instalación local de estas librerías, ya que se cargan en tiempo de ejecución desde la web.

4.3. API de OpenWeatherMap

El nodo Çlima en Pinas, EC"(`openweathermap`) requiere una clave API válida de OpenWeatherMap. Debe configurar esta clave dentro de las propiedades del nodo OpenWeatherMap en Node-RED.

5. Flujo de Datos Detallado

a) **Inicio/Despliegue:** El nodo .ªl Desplegar.ªctiva ÏNICIALIZADOR"para establecer el estado inicial de todas las variables de `flow`.

b) **Ciclo de Simulación (cada 1s):**

- "Disparador de Simulación.envía un mensaje.
- Este mensaje pasa por "Lógica de Control", que decide el estado de la bomba y las luces y envía notificaciones al "Logger de Eventos"si hay cambios.
- El mensaje continúa a "Simulador de Sensores", que actualiza todos los valores simulados de sensores y la salud de la planta, y guarda los datos para los gráficos.
- Finalmente, el mensaje llega a .ªgregador de Datos".

c) **Ciclo de Clima Real (cada 15m):**

- "Disparador de Clima.ªctiva el nodo Çlima en Pinas, EC".
- La respuesta de OpenWeatherMap es procesada por "Guarda Datos del Clima", que actualiza la variable `real_weather` en `flow`.

- Este mensaje también va al .ªgregador de Datos".

d) **Actualización del Dashboard:**

- Cada vez que .ªgregador de Datosrecibe un mensaje (ya sea de simulación o de clima), construye un `msg.payload` completo con todos los datos actuales.
- Este `msg.payload` se envía al nodo "Single Page Dashboard"(`ui_template`), que lo utiliza para actualizar toda la interfaz de usuario en tiempo real.

e) **Interacción del Usuario:**

- Cuando un usuario interactúa con un control en el dashboard (slider, botón, radio), el JavaScript en "Single Page Dashboard.ºenvía un nuevo `msg` de vuelta al flujo de Node-RED con un `topic` específico.
- "Distribuidor de Controles.ºenruta este mensaje al nodo `change` o `function` adecuado (ej., "Guardar Umbral Riego", .ªbonar Tierra").
- Estos nodos actualizan las variables de `flow` correspondientes, lo que a su vez afectará la "Lógica de Control.ºel "Simulador de Sensores.ºen el siguiente ciclo de actualización.

6. Instalación y Despliegue

a) **Instalar Node-RED:** Si aún no lo tiene, instale Node-RED en su sistema. Consulte la documentación oficial de Node-RED para su plataforma (Raspberry Pi, Docker, Windows, etc.).

b) **Instalar Nodos Adicionales:** Abra Node-RED, vaya a `Manage Palette` (Gestionar paleta) e instale `node-red-dashboard` y `node-red-node-openweathermap`.

c) **Importar el Flujo:**

- El código del flujo se encuentra en el JSON que se te proporcionó anteriormente. Puedes copiarlo directamente.
- En la interfaz de Node-RED, haga clic en el menú (tres líneas horizontales en la esquina superior derecha) ¡span class="math-inline» rightarrow¡/span¿`Import` (Importar) ¡span class="math-inline» rightarrow¡/span¿`Clipboard` (Portapapeles).
- Pegue el código JSON y haga clic en `Import`.

d) **Configurar OpenWeatherMap:**

- Haga doble clic en el nodo Çlima en Pinas, EC".
- Haga clic en el ícono del lápiz junto al campo .ªPI key"para agregar una nueva clave.
- Ingrese su clave API de OpenWeatherMap.
- Haga clic en `Add` y luego `Done`.

e) **Desplegar el Flujo:** Haga clic en el botón `Deploy` (Desplegar) en la esquina superior derecha de la interfaz de Node-RED.

f) **Acceder al Dashboard:** Una vez desplegado, puede acceder al dashboard navegando a `http://su_ip_o_dominio:1880/ui` en su navegador.

7. Consideraciones de Desarrollo y Mejoras Futuras

7.1. Simulación vs. Hardware Real

Este proyecto se basa en gran medida en la simulación de sensores. Para un huerto real, los nodos "Simulador de Sensoresz "Lógica de Control"necesitarían ser reemplazados o complementados con:

- Nodos de entrada/salida para interactuar con hardware real (ej., MQTT, GPIO, Serial, HTTP requests a microcontroladores).
- Lógica de calibración para los sensores.
- Manejo de errores y reintentos para la comunicación con el hardware.

7.2. Persistencia de Datos

Actualmente, las variables de `flow` se inicializan cada vez que el flujo se despliega o Node-RED se reinicia. Para persistir datos como el historial de gráficos o el log de eventos a largo plazo, se podría implementar:

- Almacenamiento en base de datos (ej., InfluxDB para series de tiempo, SQLite para configuraciones).
- Uso de nodos de almacenamiento de archivos.

7.3. Autenticación y Seguridad

El dashboard por defecto no tiene autenticación. Para un entorno de producción, se recomienda encarecidamente implementar:

- Autenticación de usuario para el dashboard de Node-RED.
- HTTPS para cifrar la comunicación.

7.4. Escalabilidad

Para monitorear múltiples huertos o expandir el sistema con más sensores y actuadores, se pueden considerar:

- Uso de MQTT para una comunicación más robusta y desacoplada entre dispositivos y Node-RED.

- Creación de subflujos para modularizar la lógica.
- Despliegue de instancias de Node-RED en contenedores (Docker) para facilitar la gestión.

7.5. Mejoras en la Lógica de Salud de la Planta

La simulación de la salud de la planta es rudimentaria. Se podría mejorar con:

- Modelos más complejos que consideren interacciones entre parámetros.
- Diferentes modelos de salud para diferentes tipos de plantas.
- Incorporación de más factores (ej., luz, CO2, pH).

7.6. Interfaz de Usuario

Aunque el dashboard es funcional, se pueden realizar mejoras en la UI/UX:

- Implementación de un framework frontend más avanzado (ej., Vue.js o React de forma externa a Node-RED).
- Más opciones de personalización para el usuario.
- Notificaciones push a dispositivos móviles.

8. Contacto

Para preguntas técnicas, colaboraciones o soporte, contacte a los desarrolladores del proyecto:

- Azuero Maldonado Ronald Alejandro
- Toro Ramon Diego Steeven