Implémentation des Interfaces Web

Système de Monitoring de la temperature via LoRa

17 décembre 2025

1 Architecture des Interfaces

Le système est divisé en deux interfaces web distinctes hébergées sur les ESP32 :

Structure des dossiers :

```
ProjetElectronique/
 emetteur/
    index.html    (Interface capteur)
    style.css     (Styles émetteur)
    script.js     (Logique émetteur)
     images        (Images utilisees pour designer les termes)
 recepteur/
     index.html    (Interface supervision)
     style.css     (Styles récepteur)
     script.js     (Logique récepteur)
     images        (Images utilisees pour designer les termes)
```

2 Interface Émetteur

2.1 Structure HTML (index.html)

L'interface émetteur affiche les données capteurs en temps réel :

```
<!-- Affichage des 3 capteurs -->
<div class="sensor-grid">
    <div class="sensor-card␣temperature">
        <div class="sensor-icon">         </div>
        <div class="sensor-label">Temp rature</div>
        <div id="emetteur-temp" class="sensor-value">-- C </div>
    </div>
    <!-- Cartes humidit  et fum e similaires -->
</div>

<!-- Conteneur d'alertes dynamique -->
<div id="emetteur-alert-container"></div>

<!-- Historique des mesures -->
<ul id="emetteur-history" class="history-list"></ul>
```

2.2 Styles CSS (style.css)

Palette de couleurs :

— Orange principal : `#FF8C00`

— Bleu nuit : `#1A1A2E`

— Fond blanc : `#FFFFFF`

Composants clés :

```
/* Carte capteur avec bordure color e */
.sensor-card.temperature {
    border-top: 4px solid #FF8C00;
    transition: transform 0.3s ease;
}

.sensor-card:hover {
    transform: translateY(-4px);
    box-shadow: 0 8px 24px rgba(0,0,0,0.12);
}

/* Syst me d'alerte puissant */
.alert-card {
    background: linear-gradient(135deg, #ff3d00, #dd2c00);
    animation: alertPulse 1s infinite;
    border: 4px solid #ffffff;
}

@keyframes alertPulse {
    0%, 100% { transform: scale(1); }
    50% { transform: scale(1.02); }
}

/* Message urgent clignotant */
.urgent-message {
    animation: blinkUrgent 0.8s infinite;
    font-size: 20px;
    font-weight: 800;
}

@keyframes blinkUrgent {
    0%, 100% { opacity: 1; }
    50% { opacity: 0.3; }
}
```

2.3 Logique JavaScript (script.js)

Variables d'état :

```
let tempEmetteur = 20;
let humidityEmetteur = 50;
let smokeEmetteur = 0;

let seuilTemp = 35;
let seuilHumidity = 80;
let seuilSmoke = 100;
```

Détection multi-alertes :

```
function detectAlerts(temp, humidity, smoke) {
    const alerts = [];
```

```
    if (temp >= seuilTemp) {
        alerts.push({
            type: 'temperature',
            icon: '          ',
            label: 'Alerte Temp rature',
            value: `${temp.toFixed(1)} C  (Seuil: ${seuilTemp} C )`
        });
    }
    // Logique similaire pour humidit  et fum e
    return alerts;
}
```

Affichage dynamique des alertes :

```
function displayAlerts(containerId, alerts, urgentMessage) {
    if (alerts.length === 0) {
        // Afficher  tat  normal
        return;
    }

    let alertsHTML = alerts.map(alert => `
        <div class="alert-message">
            <div class="alert-icon">${alert.icon}</div>
            <div class="alert-type-label">${alert.label}</div>
            <div class="alert-value">${alert.value}</div>
        </div>
    `).join('');

    container.innerHTML = `
        <div class="alert-card">
            <h3>       ALERTES ACTIVES (${alerts.length})</h3>
            ${alertsHTML}
            <div class="urgent-message">${urgentMessage}</div>
        </div>
    `;
}
```

Mise à jour temps réel :

```
setInterval(() => {
    // Simulation acquisition capteur DHT22
    tempEmetteur = 18 + Math.random() * 20;
    humidityEmetteur = 30 + Math.random() * 60;
    smokeEmetteur = Math.random() * 120;

    updateEmetteur();
}, 3000);
```

3 Interface Récepteur

3.1 Structure HTML (index.html)

Ajout du menu déroulant pour configuration des seuils :

```
<!-- Menu d roulant dynamique -->
<div class="threshold-dropdown">
    <div class="threshold-trigger">
        <span>Configurer les seuils d'alerte</span>
        <span class="threshold-arrow">   </span>
```

```
    </div>
    <div class="threshold-menu">
        <div class="threshold-item">
            <label>          Temp rature</label>
            <input type="number" id="seuilTemp" value="35">
            <button onclick="updateThreshold('temp')">   </button>
        </div>
        <!-- Items humidit  et fum e similaires -->
    </div>
</div>

<!-- Affichage seuils actuels -->
<div class="current-thresholds">
    <div class="threshold-display">
        <span>Temp rature:</span>
        <span id="display-temp">35 C </span>
    </div>
    <!-- Autres seuils -->
</div>
```

3.2 Styles CSS (style.css)

Menu déroulant au survol :

```
.threshold-menu {
    position: absolute;
    opacity: 0;
    visibility: hidden;
    transform: translateY(-10px);
    transition: all 0.3s ease;
}

.threshold-dropdown:hover .threshold-menu {
    opacity: 1;
    visibility: visible;
    transform: translateY(0);
}

.threshold-dropdown:hover .threshold-arrow {
    transform: rotate(180deg);
}
```

Responsive design :

```
@media (max-width: 768px) {
    .sensor-grid {
        grid-template-columns: 1fr;
    }

    .threshold-menu {
        position: relative;
        opacity: 1;
        visibility: visible;
    }
}
```

3.3 Logique JavaScript (script.js)

Gestion des seuils multiples :

```
function updateThreshold(type) {
    if (type === 'temp') {
        seuilTemp = parseFloat(
            document.getElementById('seuilTemp').value
        );
        document.getElementById('display-temp')
            .innerText = seuilTemp + ' C ';
    }
    // Logique similaire pour humidity et smoke

    updateRecepteur();
}
```

Simulation transmission LoRa :

```
// Dans setInterval de l' metteur
setTimeout(() => {
    // Ajout de bruit pour simuler transmission radio
    tempRecepteur = tempEmetteur + (Math.random()-0.5)*0.5;
    humidityRecepteur = humidityEmetteur + (Math.random()-0.5)*2;
    smokeRecepteur = smokeEmetteur + (Math.random()-0.5)*5;

    updateRecepteur();
}, 300); // Latence 300ms
```

4 Fonctionnalités Implémentées

orangepro!20 **Fonctionnalité**	**Implémentation**
Affichage temps réel	Mise à jour toutes les 3 secondes via `setInterval()`
Alertes multi-niveaux	Détection simultanée de 1, 2 ou 3 alertes
Identification précise	Chaque alerte indique type + valeur + seuil
Animations puissantes	`alertPulse`, `blinkUrgent`, `shake`
Menu déroulant	Apparition au survol avec `:hover`
Configuration seuils	3 seuils indépendants (temp, humidité, fumée)
Historique	Conservation des 50 dernières mesures
Responsive	Media queries pour mobile/tablette/desktop

TABLE 1 – Tableau des fonctionnalités

5 Animations CSS

Animations implémentées

1. `alertPulse` : Pulsation de la carte d'alerte (1s)
2. `blinkAlert` : Clignotement messages individuels (1s)
3. `shake` : Vibration icônes d'alerte (0.5s)
4. `blinkUrgent` : Clignotement message urgent (0.8s)

6 Points Techniques Clés

6.1 Détection multi-alertes simultanées

Le système gère 1, 2 ou 3 alertes en même temps grâce à un tableau dynamique :

```
const alerts = detectAlerts(temp, humidity, smoke);
// alerts.length peut  tre  0, 1, 2 ou 3
displayAlerts(containerId, alerts, message);
```

6.2 Menu déroulant sans JavaScript

L'ouverture du menu utilise uniquement CSS :

```
.threshold-dropdown:hover .threshold-menu {
    opacity: 1;
    visibility: visible;
}
```

6.3 Historique avec limitation

Les mesures sont limitées à 50 entrées pour optimiser la mémoire :

```
if (historyList.children.length > 50) {
    historyList.removeChild(historyList.lastChild);
}
```

7 Déploiement sur ESP32

1. Téléverser les fichiers via LittleFS
2. Configuration du serveur web ESP32 :

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(LittleFS, "/index.html", "text/html");
});

server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(LittleFS, "/style.css", "text/css");
});

server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(LittleFS, "/script.js", "text/javascript");
});
```

3. Accès via IP ESP32 (ex : `http://192.168.1.100`)

8 Tests Effectués

— **Navigateurs** : Chrome, Firefox, Safari, Edge
— **Résolutions** : 320px (mobile) à 1920px (desktop)
— **Performance** : Temps de chargement < 500ms
— **Alertes** : Déclenchement instantané des 3 types
— **Menu** : Ouverture fluide au survol

9 Conclusion

Les interfaces web développées offrent :

— Visualisation claire des 3 paramètres (température, humidité, fumée)
— Système d'alertes impossible à manquer (animations + couleurs vives)
— Configuration flexible des seuils (3 seuils indépendants)
— Historique complet des mesures
— Design moderne et responsive
— Code modulaire et maintenable