E-RA Widget Integration - Tech Stack & Implementation Guide

Thư Viện và Dependencies

E-RA Widget Library

```html
<script src="https://www.unpkg.com/@eohjsc/era-widget@1.1.3/src/index.js"></script>
```

- **Phiên bản**: 1.1.3
- **CDN**: unpkg.com
- **Namespace**: `@eohjsc/era-widget`

Core Architecture

1. Khởi Tạo Widget

```javascript
const eraWidget = new EraWidget();
eraWidget.init({
  needRealtimeConfigs: true,      // Cần giá trị realtime
  needHistoryConfigs: true,       // Cần giá trị lịch sử
  needActions: true,              // Cần actions (điều khiển)
  maxRealtimeConfigsCount: 3,     // Tối đa 3 config realtime
  maxHistoryConfigsCount: 1,      // Tối đa 1 config history
  maxActionsCount: 2,             // Tối đa 2 actions
  minRealtimeConfigsCount: 0,     // Tối thiểu config realtime
  minHistoryConfigsCount: 0,      // Tối thiểu config history
  minActionsCount: 0,             // Tối thiểu actions

  // Callbacks
  onConfiguration: (configuration) => {},
  onValues: (values) => {}
});
```

Data Flow Pattern

1. Configuration Object Structure

javascript

```
{
  realtime_configs: [
    { id: "config_id_1", ... },  // Config 1: Nhiệt độ
    { id: "config_id_2", ... },  // Config 2: Độ ẩm
    { id: "config_id_3", ... }   // Config 3: Trạng thái LED
  ],
  actions: [
    { action: "action_id_1", ... },  // Action 1: Bật đèn
    { action: "action_id_2", ... }   // Action 2: Tắt đèn
  ]
}
```

2. Values Object Structure

javascript

```
{
  [configId]: {
    value: actualValue,  // Giá trị thực tế
    timestamp: ...,
    ...
  }
}
```

Implementation Pattern

State Management

javascript

```
// Lưu trữ cấu hình
let configTemp = null;
let configHumi = null;
let configLed = null;
let newStatusLed = null;
let actions = [];

// Cập nhật trong onConfiguration
onConfiguration: (configuration) => {
  configTemp = configuration.realtime_configs[0];
  configHumi = configuration.realtime_configs[1];
  configLed = configuration.realtime_configs[2];
  actions = configuration.actions;
}
```

Data Binding

javascript

```
onValues: (values) => {
  // Lấy giá trị theo config ID
  const valueTemp = values[configTemp.id].value;
  const valueHumi = values[configHumi.id].value;
  const valueLed = values[configLed.id].value;

  // Cập nhật UI
  temp.innerHTML = valueTemp + ' °C';
  humi.innerHTML = valueHumi + ' %';

  // Kiểm tra thay đổi trạng thái
  if (newStatusLed !== valueLed) {
    newStatusLed = valueLed;
    statusLed.innerHTML = valueLed === 1 ? 'Đèn đang bật' : 'Đèn đang tắt';
  }
}
```

Action Triggering

javascript

```javascript
// Kích hoạt action
eraWidget.triggerAction(actions[0]?.action, null);  // Action đầu tiên
eraWidget.triggerAction(actions[1]?.action, null);  // Action thứ hai

// Gắn vào event listener
turnOn.addEventListener('click', () => {
  eraWidget.triggerAction(actions[0]?.action, null);
});
```

Best Practices

1. DOM Element References

javascript

```javascript
// Lấy element một lần khi load
const temp = document.getElementById('nhietDo');
const humi = document.getElementById('doAm');
const turnOn = document.getElementById('turnOn');
const turnOff = document.getElementById('turnOff');
const statusLed = document.getElementById('statusLed');
```

2. Null Safety

javascript

```javascript
// Sử dụng optional chaining
eraWidget.triggerAction(actions[0]?.action, null);

// Kiểm tra trước khi truy cập
if (configTemp && values[configTemp.id]) {
  const valueTemp = values[configTemp.id].value;
}
```

3. State Change Detection

javascript

```
// Lưu state cũ để so sánh
let previousLedStatus = null;

if (newStatusLed !== valueLed) {
  newStatusLed = valueLed;
  // Chỉ update UI khi có thay đổi
}
```

Integration Checklist

- [ ] Load E-RA Widget library từ CDN
- [ ] Khởi tạo `new EraWidget()`
- [ ] Cấu hình `init()` với các tham số phù hợp
- [ ] Implement `onConfiguration` callback để lưu configs
- [ ] Implement `onValues` callback để xử lý dữ liệu realtime
- [ ] Bind UI elements với data từ widget
- [ ] Setup event listeners cho actions (nếu cần điều khiển)
- [ ] Implement null safety và error handling
- [ ] Test realtime data flow
- [ ] Test action triggering

Common Use Cases

1. Sensor Data Display (Read-only)

- needRealtimeConfigs: true
- needActions: false
- Chỉ implement onValues callback

2. Control Panel (Read + Write)

- needRealtimeConfigs: true
- needActions: true
- Implement cả onValues và triggerAction

3. Historical Data View

- needHistoryConfigs: true
- Xử lý historical data trong onConfiguration

Error Handling

javascript

```
onValues: (values) => {
  try {
    if (!configTemp || !values[configTemp.id]) {
      console.warn('Temperature config not available');
      return;
    }
    const valueTemp = values[configTemp.id].value;
    temp.innerHTML = valueTemp + ' °C';
  } catch (error) {
    console.error('Error updating temperature:', error);
  }
}
```

Notes

- Widget tự động handle connection và reconnection
- Callback functions được gọi tự động khi có data mới
- Actions được trigger theo async pattern
- Config IDs được assign động bởi platform