Wolfpack Display Guide

Overview

This guide explains how to display Wolfpack inputs and outputs in rows of 4 to match the physical hardware layout.

Physical Hardware Layout

Wolfpack cards have either:
- **4 Inputs per card** OR
- **4 Outputs per card**

This means:
- Inputs 1-4 are on Card 1
- Inputs 5-8 are on Card 2
- Inputs 9-12 are on Card 3
- And so on...

Features Implemented

1. Layout Import Uses Actual Wolfpack Outputs

File: `src/app/api/ai/analyze-layout/route.ts`

The layout import API now:
- Queries actual Wolfpack outputs from the database
- Uses real output channel numbers (not arbitrary 1-N)
- Only maps TVs to configured/active outputs
- Warns if more TVs detected than available outputs

Key Changes:

```
// Queries database for actual Wolfpack outputs
const config = await prisma.matrixConfiguration.findFirst({
  where: { isActive: true },
  include: {
    outputs: {
      where: {
        isActive: true,
        status: 'active'
      },
      orderBy: { channelNumber: 'asc' }
    }
  }
})

// Uses actual output numbers
availableOutputNumbers = activeOutputs.map(output => output.channelNumber)
```

2. Matrix Display API (Rows of 4)

File: `src/app/api/matrix-display/route.ts`

New API endpoint that returns inputs/outputs formatted in rows of 4:

Endpoint: `GET /api/matrix-display`

Query Parameters:
- `includeInactive` (optional): Include inactive inputs/outputs (default: false)

Response Format:

```
{
  "inputs": {
    "total": 12,
    "rows": [
      {
        "cardNumber": 1,
        "items": [...], // Inputs 1-4
        "startChannel": 1,
        "endChannel": 4
      },
      {
        "cardNumber": 2,
        "items": [...], // Inputs 5-8
        "startChannel": 5,
        "endChannel": 8
      }
    ],
    "itemsPerRow": 4
  },
  "outputs": {
    "total": 24,
    "rows": [...],
    "itemsPerRow": 4
  },
  "metadata": {
    "totalCards": 6,
    "itemsPerCard": 4
  }
}
```

3. Display Utilities

File: `src/lib/matrix-display-utils.ts`

Utility functions for formatting and displaying inputs/outputs:

```
import { formatIntoRows, getCardNumber, getGridClasses } from '@/lib/matrix-display-utils'

// Format items into rows of 4
const rows = formatIntoRows(outputs)

// Get card number for a channel
const cardNum = getCardNumber(13) // Returns 4 (channel 13 is on card 4)

// Get CSS grid classes
const gridClasses = getGridClasses(true) // Responsive grid
```

Usage Examples

Example 1: Display Outputs in React Component

```
'use client'

import { useEffect, useState } from 'react'
import { formatIntoRows } from '@/lib/matrix-display-utils'

export default function OutputDisplay() {
  const [displayData, setDisplayData] = useState(null)

  useEffect(() => {
    fetch('/api/matrix-display')
      .then(res => res.json())
      .then(data => setDisplayData(data))
  }, [])

  if (!displayData) return <div>Loading...</div>

  return (
    <div className="space-y-8">
      <h2 className="text-2xl font-bold">Wolfpack Outputs</h2>

      {displayData.outputs.rows.map((row) => (
        <div key={row.cardNumber} className="space-y-2">
          {/* Card Label */}
          <h3 className="text-lg font-semibold text-slate-300">
            Card {row.cardNumber} - Outputs {row.startChannel}-{row.endChannel}
          </h3>

          {/* 4-column grid */}
          <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4">
            {row.items.map((output) => (
              <div
                key={output.id}
                className="bg-slate-800 rounded-lg p-4 border border-slate-700"
              >
                <div className="text-sm text-slate-400">Output {output.channelNumber}
</div>
                <div className="font-semibold text-white">{output.label}</div>
                <div className="text-xs text-slate-500 mt-2">{output.resolution}</div>
              </div>
            ))}
          </div>
        </div>
      ))}
    </div>
  )
}
```

Example 2: Display Inputs with Card Separation

```
'use client'

import { useEffect, useState } from 'react'

export default function InputDisplay() {
  const [displayData, setDisplayData] = useState(null)

  useEffect(() => {
    fetch('/api/matrix-display')
      .then(res => res.json())
      .then(data => setDisplayData(data))
  }, [])

  if (!displayData) return <div>Loading...</div>

  return (
    <div className="space-y-8">
      <h2 className="text-2xl font-bold">Wolfpack Inputs</h2>

      {displayData.inputs.rows.map((row, index) => (
        <div key={row.cardNumber}>
          {/* Card separator */}
          {index > 0 && <div className="border-t border-slate-600 my-6" />}

          {/* Card header */}
          <div className="flex items-center justify-between mb-4">
            <h3 className="text-lg font-semibold text-slate-300">
              Input Card {row.cardNumber}
            </h3>
            <span className="text-sm text-slate-500">
              Channels {row.startChannel}-{row.endChannel}
            </span>
          </div>

          {/* 4-column grid */}
          <div className="grid grid-cols-4 gap-4">
            {row.items.map((input) => (
              <div
                key={input.id}
                className="bg-slate-800 rounded-lg p-4 border border-slate-700
hover:border-blue-500 transition-colors"
              >
                <div className="text-xs text-slate-400 mb-1">
                  Input {input.channelNumber}
                </div>
                <div className="font-semibold text-white text-sm">
                  {input.label}
                </div>
                <div className="text-xs text-slate-500 mt-2">
                  {input.inputType}
                </div>
                {input.deviceType && (
                  <div className="text-xs text-blue-400 mt-1">
                    {input.deviceType}
                  </div>
                )}
              </div>
            ))}
          </div>
        </div>
      ))}
    </div>
```

```
  )
}
```

Example 3: Using Utility Functions

```
import {
  formatIntoRows,
  getCardNumber,
  getCardPosition,
  getCardChannelRange,
  calculateTotalCards
} from '@/lib/matrix-display-utils'

// Format outputs into rows
const outputs = [...] // Your outputs array
const rows = formatIntoRows(outputs)

// Get card info for a specific channel
const channel = 13
const cardNum = getCardNumber(channel) // 4
const position = getCardPosition(channel) // 1 (first position on card 4)

// Get channel range for a card
const range = getCardChannelRange(4) // { start: 13, end: 16 }

// Calculate total cards needed
const totalCards = calculateTotalCards(24) // 6 cards for 24 outputs
```

CSS Styling Recommendations

Responsive Grid (Recommended)

```
/* Mobile: 1 column */
/* Tablet: 2 columns */
/* Desktop: 4 columns */
.grid-responsive {
  display: grid;
  grid-template-columns: repeat(1, 1fr);
  gap: 1rem;
}

@media (min-width: 640px) {
  .grid-responsive {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (min-width: 1024px) {
  .grid-responsive {
    grid-template-columns: repeat(4, 1fr);
  }
}
```

Fixed 4-Column Grid

```
.grid-4-col {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 1rem;
}
```

Card Separator

```
.card-separator {
  border-top: 2px solid #475569; /* slate-600 */
  margin: 2rem 0;
}
```

Benefits

1. **Physical Hardware Match**: Display matches the actual Wolfpack card layout
2. **Easy Troubleshooting**: Users can quickly identify which card a channel is on
3. **Visual Organization**: Clear separation between cards improves readability
4. **Responsive Design**: Works on mobile, tablet, and desktop
5. **Accurate Mapping**: Layout import uses real Wolfpack outputs, not arbitrary numbers

Testing

Test the Matrix Display API

```
# Get all active inputs/outputs in rows of 4
curl http://localhost:3000/api/matrix-display

# Include inactive items
curl http://localhost:3000/api/matrix-display?includeInactive=true
```

Test Layout Import with Real Outputs

1. Configure Wolfpack outputs in the database
2. Upload a layout image/description
3. Verify the API maps TVs to actual output numbers
4. Check console logs for warnings if TVs exceed available outputs

Migration Notes

If you have existing UI components that display inputs/outputs:

1. **Update API calls**: Use `/api/matrix-display` instead of direct Prisma queries
2. **Update grid layout**: Change from arbitrary grids to 4-column layout
3. **Add card labels**: Show which card each row represents
4. **Add separators**: Visual separation between cards
5. **Update styling**: Use responsive grid classes

Future Enhancements

Potential improvements:
- Visual card indicators (different colors per card)
- Drag-and-drop reordering within cards
- Card health status indicators
- Real-time card monitoring
- Card configuration wizard