🚀 ESP32 Palletizer - Simple Vue Setup Guide

📋 Project Overview

Building a **simple but powerful** web interface for ESP32 Palletizer using:

- **Vue 3** (Composition API)
- **Shadcn/Vue** (Zero CSS framework)
- **Tailwind CSS** (Utility-first styling)
- **Vite** (Modern build tool)
- **Single HTML deployment** (ESP32 optimized)

Philosophy: Keep it Simple, Make it Work! 🎯

📁 Final Project Structure (Super Simple!)

```
esp32-palletizer-vue/
├── 📁 public/
│   └── index.html                # Base HTML template
├── 📁 src/
│   ├── 📁 components/
│   │   ├── 📁 ui/                 # Shadcn components (auto-generated)
│   │   │   ├── button.vue
│   │   │   ├── card.vue
│   │   │   ├── input.vue
│   │   │   ├── slider.vue
│   │   │   ├── badge.vue
│   │   │   └── textarea.vue
│   │   ├── AppHeader.vue         # Header + Status + Theme Toggle
│   │   ├── SystemControls.vue    # PLAY, PAUSE, STOP, IDLE, ZERO buttons
│   │   ├── SpeedControls.vue     # Speed matrix untuk semua axis
│   │   └── CommandEditor.vue     # Command editor + file upload
│   ├── 📁 composables/
│   │   ├── useApi.js            # API calls ke ESP32 (/command, /status,
dll)
│   │   ├── useTheme.js          # Dark mode toggle & persistence
│   │   └── useWebSocket.js      # Real-time status updates via EventSource
│   ├── 📁 lib/
│   │   └── utils.js             # Shadcn utilities (auto-generated)
│   ├── 📁 assets/
│   │   └── index.css            # Global CSS + Tailwind + Design tokens
│   ├── App.vue                  # Main app - layout semua components
│   └── main.js                  # Entry point
├── build-esp32.js               # ESP32 single-file build script
├── components.json              # Shadcn configuration
├── tailwind.config.js           # Tailwind configuration
├── vite.config.js               # Vite build configuration
├── package.json                 # Dependencies & scripts
└── README.md                    # Project documentation
```

Total Files: ~15 files aja! (vs ~50+ files di struktur complex)

🛠️ Step 1: Create Vue Project

bash

```bash
# Create Vue project
npm create vue@latest esp32-palletizer-vue

# Configuration - Keep it SIMPLE:
✅ Project name: esp32-palletizer-vue
❌ TypeScript? No
❌ JSX Support? No
❌ Vue Router? No (single page)
❌ Pinia? No (use composables)
❌ Vitest? No
❌ End-to-End Testing? No
✅ ESLint? Yes
✅ Prettier? Yes

# Navigate to project
cd esp32-palletizer-vue

# Install base dependencies
npm install
```

🎨 Step 2: Setup Tailwind CSS

bash

```bash
# Install Tailwind
npm install -D tailwindcss postcss autoprefixer @tailwindcss/forms
npx tailwindcss init -p
```

tailwind.config.js:

javascript

```javascript
/** @type {import('tailwindcss').Config} */
export default {
  darkMode: ['class'],
  content: [
    './index.html',
    './src/**/*.{vue,js}',
  ],
  theme: {
    extend: {
      colors: {
        border: 'hsl(var(--border))',
        input: 'hsl(var(--input))',
        ring: 'hsl(var(--ring))',
        background: 'hsl(var(--background))',
        foreground: 'hsl(var(--foreground))',
        primary: {
          DEFAULT: 'hsl(var(--primary))',
          foreground: 'hsl(var(--primary-foreground))'
        },
        secondary: {
          DEFAULT: 'hsl(var(--secondary))',
          foreground: 'hsl(var(--secondary-foreground))'
        },
        destructive: {
          DEFAULT: 'hsl(var(--destructive))',
          foreground: 'hsl(var(--destructive-foreground))'
        },
        muted: {
          DEFAULT: 'hsl(var(--muted))',
          foreground: 'hsl(var(--muted-foreground))'
        },
        accent: {
          DEFAULT: 'hsl(var(--accent))',
          foreground: 'hsl(var(--accent-foreground))'
        },
        card: {
          DEFAULT: 'hsl(var(--card))',
          foreground: 'hsl(var(--card-foreground))'
        }
      },
      borderRadius: {
        lg: 'var(--radius)',
        md: 'calc(var(--radius) - 2px)',
        sm: 'calc(var(--radius) - 4px)'
      }
    }
  },
  plugins: [require('@tailwindcss/forms')]
}
```

🧩 Step 3: Setup Shadcn/Vue

bash

```bash
# Install Shadcn/Vue
npx shadcn-vue@latest init

# Configuration:
❌ TypeScript? No
✅ Framework: Vite
✅ Style: Default
✅ Base color: Blue
✅ Global CSS: src/assets/index.css
✅ CSS variables: Yes
✅ Tailwind config: tailwind.config.js
✅ Components alias: src/components
✅ Utils alias: src/lib/utils
```

Install hanya components yang dibutuhkan:

bash

```bash
# Core components untuk Palletizer
npx shadcn-vue@latest add button
npx shadcn-vue@latest add card
npx shadcn-vue@latest add input
npx shadcn-vue@latest add slider
npx shadcn-vue@latest add badge
npx shadcn-vue@latest add textarea
```

⚙️ Step 4: Configure Vite for ESP32

vite.config.js:

```javascript
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import path from 'path'

export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src')
    }
  },
  build: {
    // ESP32 optimization
    target: 'es2015',
    minify: 'terser',
    rollupOptions: {
      output: {
        inlineDynamicImports: true,
        manualChunks: undefined
      }
    },
    chunkSizeWarningLimit: 1500
  },
  server: {
    port: 3000
  }
})
```

🔧 Step 5: ESP32 Build Script

build-esp32.js:

javascript

```javascript
const fs = require('fs')
const path = require('path')

console.log('🚀 Building single HTML for ESP32...')

const distPath = path.join(__dirname, 'dist')
const htmlPath = path.join(distPath, 'index.html')

// Read HTML file
let html = fs.readFileSync(htmlPath, 'utf8')

// Inline CSS files
const assetsPath = path.join(distPath, 'assets')
if (fs.existsSync(assetsPath)) {
  const files = fs.readdirSync(assetsPath)

  // Inline CSS
  files.filter(f => f.endsWith('.css')).forEach(cssFile => {
    const cssContent = fs.readFileSync(path.join(assetsPath, cssFile), 'utf8')
    html = html.replace(
      new RegExp(`<link[^>]*href="[^"]*${cssFile}"[^>]*>`, 'g'),
      `<style>${cssContent}</style>`
    )
  })

  // Inline JS
  files.filter(f => f.endsWith('.js')).forEach(jsFile => {
    const jsContent = fs.readFileSync(path.join(assetsPath, jsFile), 'utf8')
    html = html.replace(
      new RegExp(`<script[^>]*src="[^"]*${jsFile}"[^>]*></script>`, 'g'),
      `<script>${jsContent}</script>`
    )
  })
}

// Write inlined HTML
fs.writeFileSync(htmlPath, html)

// Clean assets folder
if (fs.existsSync(assetsPath)) {
  fs.rmSync(assetsPath, { recursive: true })
}

const sizeKB = (fs.statSync(htmlPath).size / 1024).toFixed(1)
console.log(`✅ Done! File: dist/index.html (${sizeKB} KB)`)
```

📦 Step 6: Package.json Scripts

Add ke package.json:

json

```json
{
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "build:esp32": "vite build && node build-esp32.js",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.4.0",
    "@vueuse/core": "^10.7.0"
  }
}
```

Install tambahan:

bash

```bash
npm install @vueuse/core
```

🎨 Step 7: Global Styles

src/assets/index.css:

css

```css
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 222.2 84% 4.9%;
    --card: 0 0% 100%;
    --card-foreground: 222.2 84% 4.9%;
    --primary: 221.2 83.2% 53.3%;
    --primary-foreground: 210 40% 98%;
    --secondary: 210 40% 96%;
    --secondary-foreground: 222.2 84% 4.9%;
    --muted: 210 40% 96%;
    --muted-foreground: 215.4 16.3% 46.9%;
    --accent: 210 40% 96%;
    --accent-foreground: 222.2 84% 4.9%;
    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 210 40% 98%;
    --border: 214.3 31.8% 91.4%;
    --input: 214.3 31.8% 91.4%;
    --ring: 221.2 83.2% 53.3%;
    --radius: 0.75rem;
  }

  .dark {
    --background: 222.2 84% 4.9%;
    --foreground: 210 40% 98%;
    --card: 222.2 84% 4.9%;
    --card-foreground: 210 40% 98%;
    --primary: 210 40% 98%;
    --primary-foreground: 222.2 84% 4.9%;
    --secondary: 217.2 32.6% 17.5%;
    --secondary-foreground: 210 40% 98%;
    --muted: 217.2 32.6% 17.5%;
    --muted-foreground: 215 20.2% 65.1%;
    --accent: 217.2 32.6% 17.5%;
    --accent-foreground: 210 40% 98%;
    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 40% 98%;
    --border: 217.2 32.6% 17.5%;
    --input: 217.2 32.6% 17.5%;
    --ring: 212.7 26.8% 83.9%;
  }

  * {
    @apply border-border;
  }

  body {
    @apply bg-background text-foreground;
  }
}
```

```
/* Industrial status indicators */
@layer components {
  .status-running { @apply bg-green-100 text-green-800 dark:bg-green-900 dark:te
  .status-idle { @apply bg-gray-100 text-gray-800 dark:bg-gray-800 dark:text-gra
  .status-paused { @apply bg-yellow-100 text-yellow-800 dark:bg-yellow-900 dark:
  .status-error { @apply bg-red-100 text-red-800 dark:bg-red-900 dark:text-red-3
}
```

🎛 Step 8: Create Core Files

Entry Point

src/main.js:

```javascript
import { createApp } from 'vue'
import App from './App.vue'
import './assets/index.css'

createApp(App).mount('#app')
```

Main App Layout

src/App.vue:

```vue
<template>
  <div class="min-h-screen bg-background">
    <!-- Header -->
    <AppHeader />

    <!-- Main Content -->
    <div class="container mx-auto p-4 space-y-6 max-w-6xl">
      <!-- Control Row -->
      <div class="grid grid-cols-1 lg:grid-cols-2 gap-6">
        <SystemControls />
        <SpeedControls />
      </div>

      <!-- Command Editor -->
      <CommandEditor />
    </div>
  </div>
</template>

<script setup>
import AppHeader from '@/components/AppHeader.vue'
import SystemControls from '@/components/SystemControls.vue'
import SpeedControls from '@/components/SpeedControls.vue'
import CommandEditor from '@/components/CommandEditor.vue'
</script>
```

Create Placeholder Components

src/components/AppHeader.vue:

vue

```vue
<template>
  <header class="bg-card border-b border-border">
    <div class="container mx-auto px-4 py-4 flex justify-between items-center ma
      <div class="flex items-center space-x-3">
        <span class="text-2xl">🤖</span>
        <h1 class="text-xl font-bold">ESP32 Palletizer</h1>
      </div>

      <div class="flex items-center space-x-4">
        <!-- Status Badge -->
        <Badge :class="statusClass">{{ status }}</Badge>

        <!-- Theme Toggle -->
        <Button variant="ghost" size="sm" @click="toggleTheme">
          {{ isDark ? '🌙' : '☀️' }}
        </Button>
      </div>
    </div>
  </header>
</template>

<script setup>
import { computed } from 'vue'
import { Button } from '@/components/ui/button'
import { Badge } from '@/components/ui/badge'
import { useTheme } from '@/composables/useTheme'

// Placeholder - akan diganti dengan real data
const status = 'IDLE'

const { isDark, toggleTheme } = useTheme()

const statusClass = computed(() => {
  const base = 'px-2 py-1 text-xs font-medium rounded-full'
  switch (status) {
    case 'RUNNING': return `${base} status-running`
    case 'PAUSED': return `${base} status-paused`
    case 'IDLE': return `${base} status-idle`
    default: return `${base} status-error`
  }
})
</script>
```

src/components/SystemControls.vue:

vue

```vue
<template>
  <Card>
    <CardHeader>
      <CardTitle class="flex items-center space-x-2">
        <span>🎮</span>
        <span>System Controls</span>
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div class="grid grid-cols-2 gap-3">
        <Button class="bg-green-600 hover:bg-green-700">▶ PLAY</Button>
        <Button variant="outline" class="border-yellow-500 text-yellow-600">⏸ P
        <Button variant="destructive">⏹ STOP</Button>
        <Button variant="secondary">🟣 IDLE</Button>
        <Button class="col-span-2 bg-purple-600 hover:bg-purple-700">⌂ ZERO</But
      </div>
    </CardContent>
  </Card>
</template>

<script setup>
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card'
import { Button } from '@/components/ui/button'
</script>
```

src/components/SpeedControls.vue:

vue

```vue
<template>
  <Card>
    <CardHeader>
      <CardTitle class="flex items-center space-x-2">
        <span>⚡</span>
        <span>Speed Controls</span>
      </CardTitle>
    </CardHeader>
    <CardContent class="space-y-4">
      <!-- Speed controls placeholder -->
      <div class="text-center text-muted-foreground">
        Speed matrix will go here
      </div>
    </CardContent>
  </Card>
</template>

<script setup>
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card'
</script>
```

src/components/CommandEditor.vue:

vue

```
<template>
  <Card>
    <CardHeader>
      <CardTitle class="flex items-center space-x-2">
        <span>📝</span>
        <span>Command Editor</span>
      </CardTitle>
    </CardHeader>
    <CardContent>
      <Textarea
        placeholder="Enter commands here..."
        class="min-h-[200px] font-mono text-sm"
      />
      <div class="flex space-x-2 mt-4">
        <Button>💾 Save</Button>
        <Button variant="outline">📤 Upload</Button>
        <Button variant="outline">📥 Download</Button>
      </div>
    </CardContent>
  </Card>
</template>

<script setup>
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card'
import { Button } from '@/components/ui/button'
import { Textarea } from '@/components/ui/textarea'
</script>
```

Create Basic Composables

src/composables/useTheme.js:

```javascript
import { ref, watch } from 'vue'
import { useLocalStorage } from '@vueuse/core'

export function useTheme() {
  const isDark = useLocalStorage('palletizer-theme', false)

  const toggleTheme = () => {
    isDark.value = !isDark.value
  }

  // Apply theme to document
  watch(isDark, (dark) => {
    if (dark) {
      document.documentElement.classList.add('dark')
    } else {
      document.documentElement.classList.remove('dark')
    }
  }, { immediate: true })

  return {
    isDark,
    toggleTheme
  }
}
```

src/composables/useApi.js:

javascript

```javascript
import { ref } from 'vue'

export function useApi() {
  const loading = ref(false)
  const error = ref(null)

  const sendCommand = async (cmd) => {
    loading.value = true
    error.value = null

    try {
      const response = await fetch('/command', {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: `cmd=${cmd}`
      })

      if (!response.ok) throw new Error('Command failed')
      return await response.text()
    } catch (err) {
      error.value = err.message
      throw err
    } finally {
      loading.value = false
    }
  }

  return {
    loading,
    error,
    sendCommand
  }
}
```

src/composables/useWebSocket.js:

javascript

```javascript
import { ref, onMounted, onUnmounted } from 'vue'

export function useWebSocket() {
  const status = ref('IDLE')
  const connected = ref(false)

  let eventSource = null

  const connect = () => {
    eventSource = new EventSource('/events')

    eventSource.onopen = () => {
      connected.value = true
    }

    eventSource.onmessage = (event) => {
      try {
        const data = JSON.parse(event.data)
        if (data.type === 'status') {
          status.value = data.value
        }
      } catch (err) {
        console.error('WebSocket parse error:', err)
      }
    }

    eventSource.onerror = () => {
      connected.value = false
    }
  }

  const disconnect = () => {
    if (eventSource) {
      eventSource.close()
      eventSource = null
    }
    connected.value = false
  }

  onMounted(connect)
  onUnmounted(disconnect)

  return {
    status,
    connected
  }
}
```

🧪 Step 9: Test Setup

bash

```bash
# Install dependencies (if not done)
npm install

# Start development server
npm run dev
# Should open http://localhost:3000

# Build for ESP32
npm run build:esp32
# Should create dist/index.html (~50-80KB)
```

Expected Result:

- ✅ App loads with header, system controls, speed controls, command editor
- ✅ Dark mode toggle works
- ✅ Professional industrial UI
- ✅ Single HTML file generated for ESP32

🚀 Step 10: Development Workflow

bash

```bash
# Development
npm run dev                # Hot reload development

# Production build
npm run build:esp32        # Single HTML for ESP32

# Code quality
npm run lint               # Fix code issues
npx prettier --write src # Format code
```

ESP32 Deployment

1. Run `npm run build:esp32`
2. Copy `dist/index.html` content
3. Upload to ESP32 LittleFS as `/index.html`
4. Access via ESP32 IP address

🎯 What's Next?

After this setup, you'll implement:

1. **Real API integration** in composables
2. **System controls functionality**
3. **Speed matrix with sliders**
4. **Command editor with file upload**
5. **Real-time status updates**

✅ Final Checklist

- [ ] Vue project created
- [ ] Shadcn/Vue installed with required components
- [ ] ESP32 build script working
- [ ] Basic layout with 4 main components
- [ ] Theme toggle working
- [ ] Single HTML output < 100KB
- [ ] Development server running

🎉 **Simple setup complete! Ready to build the actual functionality!**