1. What Is LEDSpicer?

LEDSpicer is an open-source LED animation and control program that provides game-aware control of LED hardware in arcade cabinets. If your buttons have LED's in them - either RGB or single color - this program will let you control them. LEDSpicer can also control things like joystick restrictors for 4/8 way use. For a full list of functionality, see the LEDSpicer github page.

There are a couple of configuration options that complicate this enormously. The build of LEDSPicer is now correct, and this information applies to the next **butterfly** release.

1.1 What It Does

At its core, LEDSpicer acts as a mediator between your LED hardware and the software running on the machine. It:

- **Manages LED hardware** through device plugins.
- **Animates LEDs** using a library of built-in effects: pulse, gradient, serpentine, random, filler, and audio-reactive modes driven by PulseAudio or ALSA.
- **Loads profiles** automatically when EmulationStation (or another frontend) signals that a game has been selected or launched.
- **Responds to MAME output events** in real time, so a button can light up the moment it becomes active in-game.
- **Transitions between profiles** using effects like FadeOutIn, Crossfade, or Curtain wipes.

1.2 Architecture

The daemon (`ledspicerd`) runs as a background service. A companion utility called `emitter` is used by shell scripts to send commands to the running daemon — telling it to load a profile, finish the current one, or fire a one-shot animation.

LEDSpicer has three plugin categories built on top:

Plugin Type	Purpose
Device plugins	Communicate with specific hardware boards over USB
Animation plugins	Generate LED effects frame by frame
Input plugins	Listen for external triggers (MAME events, network commands, EmulationStation scripts)

1.3 Supported Hardware

LEDSpicer natively supports:

- **Ultimarc**: Ultimate I/O, PacLED64, NanoLed, PAC Drive
- **GroovyGameGear**: Led-Wiz 32
- **WolfWare**: Howler
- **Adalight** (serial/USB LED strips)

- **Raspberry** Pi GPIO

For input control (joystick restrictors, spinner interfaces), it also supports the GP-Wiz40 RotoX, ServoStik, UltraStik360, and TOS GRS Gate Restrictor.

LEDSpicer has it's own nomenclature, and I'll try to explain it as I go along.

The hardware used in this tutorial:

1. Ultimarc Ultimate I/O
2. Two player, eight buttons each
3. Two LED joysticks
4. Two start buttons
5. Two coin buttons
6. Four utility buttons
7. LED Trackball
8. In Keyboard mode configured with keyboardToPads in Batocera

This is not a comprehensive example. There are many different ways to configure this for use with Batocera; this is just one way.

---

2. LEDSpicer in Batocera

2.1 Enabling the Service

Don't do this until you've gone through this **entire** document.

1. Boot Batocera and navigate to **Main Menu → System Settings → Services**.
2. Locate **LEDSpicer** and enable it.
3. Restart Batocera.

From the command line, run the commands:

```
batocera-services enable ledspicer
batocera-services start ledspicer
```

A restart is not needed in this case.

2.2 Files and layout

```
/userdata/system/configs/ledspicer/
├── ledspicer.conf
├── basicColors.xml
├── colors.ini
├── controls.ini
```

```
├── gameData.xml
├── profiles/
│   ├── default.xml
│   ├── arcade.xml
│   ├── JOYSTICK2_B1.xml
│   └── arcade/
│       ├── mslug.xml
│       └── sf2.xml
├── inputs/
├── umaps/
└── animations/
```

Scripts that notify LEDSpicer of game events are placed in:

```
/userdata/system/configs/emulationstation/scripts/
├── game-start/
├── game-end/
├── game-selected/
└── system-selected/
```

---

3. Configuration Files in Detail

All LEDSpicer configuration is XML. The files and value are case-sensitive.

3.1 ledspicer.conf — The Main Daemon Configuration

This is the most important file. It tells the daemon what hardware is attached, what LEDs exist, how they are grouped, and global runtime parameters.

3.1.1 Root Element

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer
    version="1.1"
    type="Configuration"
    fps="30"
    port="16161"
    colors="basicColors"
    logLevel="Warning"
    userId="1000"
    groupId="1000"
    craftProfile="True"
    dataSource="controls.ini"
>
```

Attribute	Purpose
`colors`	Name of the colors definition file (no extension).
`dataSource`	`MAME`, `controls.ini`, `file` Where to find information on what controls a game uses.
	`MAME`: Asks the running MAME instance.
	`controls.ini`: Parses this file in the `config` directory.
	`file`: Parses the file `gameData.xml` in the `config` directory. Currently broken.
`fps`	Animation frames per second. 14–30 is typical.
`logLevel`	`Error`, `Warning`, `Info`, or `Debug`.
`port`	TCP port `emitter` uses to talk to the daemon. Default is `16161`.

3.1.2 The <devices> Section

This declares the physical hardware. For a single Ultimate IO board:

```
<devices>
    <device name="UltimarcUltimate" boardId="1">
        <!-- element definitions go here -->
    </device>
</devices>
```

`boardId` distinguishes multiple boards of the same type if you have more than one.

3.1.3 Defining Elements

Elements are the named, logical representations of individual LEDs or RGB triplets on your control panel. The naming convention `P[player]_[type][number]` is strongly recommended because LEDSpicer uses it when auto-matching profiles to games.

Single-color LED (one pin):

```
<element name="P1_COIN" led="3"/>
```

RGB LED (three pins — red, green, blue):

```
<element name="P1_BUTTON1" red="11" green="12" blue="13"/>
<element name="P1_BUTTON2" red="14" green="15" blue="16"/>
<element name="P1_BUTTON3" red="17" green="18" blue="19"/>
<element name="P1_BUTTON4" red="20" green="21" blue="22"/>
<element name="P1_BUTTON5" red="23" green="24" blue="25"/>
<element name="P1_BUTTON6" red="26" green="27" blue="28"/>
<element name="P1_JOYSTICK" red="5"  green="6"  blue="7"/>
<element name="P1_TRACKBALL" red="41"  green="42"  blue="43"/>
<element name="P1_START"    red="8"  green="9"  blue="10"/>

<element name="P2_BUTTON1" red="41" green="42" blue="43"/>
<element name="P2_BUTTON2" red="44" green="45" blue="46"/>
<!-- ... and so on ... -->
<element name="P2_JOYSTICK" red="35" green="36" blue="37"/>
```

```
<element name="P2_START"    red="38" green="39" blue="40"/>
```

Pin numbering: The Ultimate IO exposes 96 pins (1–96). Each RGB LED consumes three consecutive pins. Map these according to how your harness is physically wired to the board. See the *__Troubleshooting section__* for how to determine or validate your pin settings.

3.1.4 The <layout> Section

The layout organizes elements into named groups. Groups are what animations and profiles operate on. A special group called `All` is generated automatically from every element in the devices section.

```
<layout defaultProfile="default">

    <group name="Player1Buttons">
        <element name="P1_BUTTON1"/>
        <element name="P1_BUTTON2"/>
        <element name="P1_BUTTON3"/>
        <element name="P1_BUTTON4"/>
        <element name="P1_BUTTON5"/>
        <element name="P1_BUTTON6"/>
    </group>

    <group name="Player2Buttons">
        <element name="P2_BUTTON1"/>
        <element name="P2_BUTTON2"/>
        <!-- ... -->
    </group>

    <group name="Joysticks">
        <element name="P1_JOYSTICK"/>
        <element name="P2_JOYSTICK"/>
    </group>

    <group name="StartButtons">
        <element name="P1_START"/>
        <element name="P2_START"/>
    </group>

</layout>
```

The `led` attribute inside a group is a **position index** used by animations for sequencing, not the physical pin number — the physical mapping was already declared in the `<devices>` section.

`defaultProfile` sets which profile loads when no game-specific profile is found.

3.2 Colors Files

This files defines the names and hex values of the colors used in `profiles`.

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer version="1.1" type="Colors" format="hex"
>
  <color name="White" color="FFFFFF" />
  <color name="LightGray" color="D3D3D3" />
  <!-- ... -->
  <color name="DarkOrange" color="8B4000" />
  <color name="VeryDarkOrange" color="A0522D" />
</LEDSpicer>
```

You can define as few or as many colors here as you want. A couple of full examples will be linked to.

3.3 Profile Files

Each profile is an XML file stored in the `profiles/` subdirectory. A profile defines what the LEDs look like during a particular game or system state.

3.3.1 Default Profile (default.xml)

This is the attract-mode or idle profile — shown when browsing the game list.

Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer version="1.1" type="Profile" backgroundColor="Black">

  <!-- Fade transition when switching away from this profile -->
  <transition name="FadeOutIn" speed="Normal" color="Off"/>

  <animations>
    <animation name="IdleGradient" />
  </animations>

</LEDSpicer>
```

Example 2

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer version="1.1" type="Profile" backgroundColor="Black">

    <animations>
        <animation name="Pulse" group="All">
```

```
            <parameter name="color" value="Blue"/>
            <parameter name="speed" value="Medium"/>
        </animation>
    </animations>

    <alwaysOnElements>
        <element name="P1_COIN" color="White"/>
        <element name="P2_COIN" color="White"/>
    </alwaysOnElements>

</LEDSpicer>
```

3.3.2 Game-Specific Profile

For a fighting game that uses 6 buttons per player, a profile might look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer version="1.1" type="Profile" backgroundColor="Black">

    <alwaysOnElements>
        <element name="P1_BUTTON1" color="Red"/>
        <element name="P1_BUTTON2" color="Yellow"/>
        <element name="P1_BUTTON3" color="Blue"/>
        <element name="P1_BUTTON4" color="Green"/>
        <element name="P1_BUTTON5" color="White"/>
        <element name="P1_BUTTON6" color="Purple"/>
        <element name="P1_START"   color="White"/>
        <element name="P2_BUTTON1" color="Red"/>
        <!-- ... -->
    </alwaysOnElements>

</LEDSpicer>
```

LEDSpicer searches for profiles in this priority order when a game launches:

1. A profile named exactly after the ROM file (`sf2.xml`) located in the `profiles/arcade/` directory.
2. A profile named after the player/button count (e.g., `P2_B1.xml` for games with two players, each with one button) located in the `profiles/` directory.
3. A profile named after the controller/button count (e.g., `JOYSTICK2_B1` for games with two joysticks, each with one button) located in the `profiles/` directory.
4. A profile named after the platform (`arcade.xml`) located in the `profiles/` directory.
5. The `default.xml` fallback

This cascading lookup means you only need to create profiles for games that need special treatment; everything else falls back gracefully.

3.4 Animation Files

Just as you can have an arbitrary number of profiles, you can also have an arbitrary number of animation definitions. Animations are referenced in `profile` files like this:

```
  <animations>
    <animation name="IdleGradient" />
  </animations>
```

LEDSpicer would look for this animation at `animations/IdleGradient.xml`.

3.4.1 First example

The first example would cause the gradient to run across all mapped elements from your configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LEDSpicer version="1.1" type="Animation">

  <!-- Rainbow gradient cycling across all buttons -->
  <actor
    type="Gradient"
    group="Panel"
    filter="Normal"
    speed="Normal"
    direction="Forward"
    colors="Red,Orange,Yellow,Green,Cyan,Blue,Violet,Magenta"
    mode="Cyclic"
    tones="20"
  />

</LEDSpicer>
```

3.4.1 Second example

This example plays the same gradient for both player one and player two, but out-of-step. It also has seperate effects for the coin and start buttons.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Attract / idle animation.
  Two layers:
    1. Cyclic rainbow gradient sweeps across all action buttons.
    2. Slower pulse on Start/Coin buttons so they breathe in their default
colors.
-->
<LEDSpicer version="1.1" type="Animation">
```

```
  <!-- Rainbow gradient cycling across Player 1 buttons -->
  <actor
    type="Gradient"
    group="P1"
    filter="Normal"
    speed="Normal"
    direction="Forward"
    colors="Red,Orange,Yellow,Green,Cyan,Blue,Violet,Magenta"
    mode="Cyclic"
    tones="20"
  />

  <!-- Same gradient on Player 2 buttons, offset by starting further into
the cycle -->
  <actor
    type="Gradient"
    group="P2"
    filter="Normal"
    speed="Normal"
    direction="Forward"
    colors="Blue,Violet,Magenta,Red,Orange,Yellow,Green,Cyan"
    mode="Cyclic"
    tones="20"
  />

  <!-- Start buttons breathe green -->
  <actor
    type="Pulse"
    group="UTILITY_BUTTONS"
    filter="Normal"
    speed="Slow"
    direction="Forward"
    color="Green"
  />

  <!-- Coin buttons breathe yellow -->
  <actor
    type="Pulse"
    group="COIN_BUTTONS"
    filter="Normal"
    speed="Slow"
    direction="Forward"
    color="Yellow"
  />

</LEDSpicer>
```

3.5 EmulationStation Integration Scripts

Batocera's EmulationStation calls scripts at four lifecycle events. Each script sends an `emitter` command to the running daemon.

Game-Start Script
`/userdata/system/configs/emulationstation/scripts/game-start/ledspicer.sh`

```
#!/usr/bin/env bash

romPath="${1}"
romName="${2}"

system=$(echo "$romPath" | awk -F'/roms/' '{print $2}' | cut -d'/' -f1)

if grep -q "\\[$romName\\]" /userdata/system/configs/ledspicer/colors.ini;
  then system="arcade";
fi

emitter -r LoadProfileByEmulator "$romName" "$system"
```

This is where a profile is loaded when a rom is played. You will want this script executable.

Game-End Script
`/userdata/system/configs/emulationstation/scripts/game-end/ledspicer.sh`

```
#!/usr/bin/env bash
rotator 1 1 8
/usr/bin/emitter FinishLastProfile
```

This is run when a rom is quit. You will want this script executable.

Game-Selected Script
`/userdata/system/configs/emulationstation/scripts/game-selected/ledspicer.sh`

```
#!/usr/bin/env bash

system="${1}"
romPath="${2}"

romPath=$(basename "${romPath}")
romName="${romPath%.*}"

if grep -q "\\[$romName\\]" /userdata/system/configs/ledspicer/colors.ini;
  then system="arcade";
fi

emitter -r -n -f "NO_INPUTS|SHOW_ROTATOR" LoadProfileByEmulator "$romName"
"$system"
```

This is where a profile is loaded when a rom is selected; a rom is selected either when scrolling through the game list of a system, or when the screen saver is running. As this will cause profiles to be loaded rapidly, you will not typically want this script executable.

System-Selected Script
`/userdata/system/configs/emulationstation/scripts/system-selected/ledspicer.sh`

```
#!/usr/bin/env bash
system="$1"
/usr/bin/emitter LoadProfile "$system"
```

This is where a profile is loaded when a system is selected; a system is selected either when scrolling through the systems list of a system, or when the screen saver is running. As this will cause profiles to be loaded rapidly, you will not typically want this script executable.

Making Scripts Executable

After placing the scripts, apply execute permissions via SSH only to the events you want LEDSpicer to respond to:

```
chmod +x /userdata/system/configs/emulationstation/scripts/game-start/ledspicer.sh
chmod +x /userdata/system/configs/emulationstation/scripts/game-end/ledspicer.sh
chmod +x /userdata/system/configs/emulationstation/scripts/game-selected/ledspicer.sh
chmod +x /userdata/system/configs/emulationstation/scripts/system-selected/ledspicer.sh
```

If you later change your mind, simply remove the execute permission:

```
chmod -x /userdata/system/configs/emulationstation/scripts/game-selected/ledspicer.sh
chmod -x /userdata/system/configs/emulationstation/scripts/system-selected/ledspicer.sh
```

3.6 MAME Input Plugin (Real-Time Button Events)

For MAME games, LEDSpicer can go further and light buttons reactively — illuminating only the buttons the current game actually uses, based on MAME's output event system.

In the profile's `<inputs>` section, reference the MAME input plugin:

```
<inputs>
    <input type="Mame" mapFile="mame.ini"/>
</inputs>
```

The map file (`mame.ini`) connects MAME output event names to element names and colors:

```
[P1_B1]
target = P1_BUTTON1
color  = Red
filter = Normal

[P1_B2]
target = P1_BUTTON2
color  = Yellow
filter = Normal
```

This requires MAME to be configured to emit output events, which is enabled in `mame.ini` with `output = network`.

---

4. Testing & Troubleshooting

Step 4.1: Confirm wiring and configuration of LEDs

Step 4.1.1: Confirm wiring of LEDs

If you are not sure of the wiring of the LEDs, over SSH, you can run this command:

```
ledspicerd -l
```

This will produce output like:

```
Set interval to 33ms
Setting FPS to 30
Reading /userdata/system/configs/ledspicer/basicColors.xml
Loading library /usr/lib/ledspicer/devices/UltimarcUltimate.so
Opening USB session
Processing Ultimarc Ipac Ultimate IO Id: 1
Ultimarc Ipac Ultimate IO Id: 1 with 96 LEDs divided into 23 elements
LED 31 is not set for Ultimarc Ipac Ultimate IO Id: 1
LED 32 is not set for Ultimarc Ipac Ultimate IO Id: 1
LED 33 is not set for Ultimarc Ipac Ultimate IO Id: 1
...
Initializing Device Ultimarc Ipac Ultimate IO Id: 1
Connecting to 0xd209:0x410 Id: 1
Detecting interface
No Game Controller mode detected
Claiming interface 2
Dropping privileges to user id 1000 and group id 1000
Privileges dropped
Reading /userdata/system/configs/ledspicer/profiles/main.xml
```

```
Reading /userdata/system/configs/ledspicer/animations/IdleGradient.xml
Test LEDs (q to quit)

Select a Led (r to reset, q to quit):
```

This shows that 23 elements have been configured, and also that particular pins are not mapped to any buttons. All buttons should be off (although sometimes not). To see what button pins are mapped to, simply start by entering 1 and hitting return. If an LED illuminates, note which one it is and update `ledspicer.conf`.

For example, if this causes player 1 button one to illuminate, we can update this line in our `ledspicer.conf` file:

```
<element name="P1_BUTTON1" red="10" green="11" blue="12" />
```

to:

```
<element name="P1_BUTTON1" red="1" green="2" blue="3" />
```

As the RGB LEDs are mapped to three adjacent pins, this works. Next we would enter 4 and hit return. If no LED illuminates, we can add 3 to the number and try again. Do this until all the LEDs are mapped. When finished, `r` return then `q` return to exit back to the command line.

If you have single color LEDs, simply iterate by 1 instead of 3.

Step 4.1.2: Confirm wiring of LEDs

To validate you have configured each element correctly use this command:

```
ledspicerd -e
```

The default Ultimarc color animation may play; ignore this.

The output will be something like:

```
Enter
P2_BUTTON7
P2_BUTTON6
P2_BUTTON5
Pause
P2_BUTTON4
P2_BUTTON3
P1_BUTTON2
P1_BUTTON7
P1_BUTTON3
P1_TRACKBALL
P2_JOYSTICK
P1_JOYSTICK
P1_BUTTON4
```

```
P1_BUTTON1
P1_BUTTON5
P1_BUTTON6
P1_CREDIT
P2_BUTTON8
P1_BUTTON8
P2_BUTTON1
Tab
P2_BUTTON2

Enter an element name (r to reset, q to quit):
```

This lists all of the elements defined in your configuration file. To test an element, type its name and hit enter. That element should illuminate. Again, if the default Ultimarc animation is playing, once the animation gets to that element, it will reset. But until then it should be white.

Step 4.2: Test the Daemon

Over SSH, start the daemon in debug mode to verify configuration:

```
ledspicerd -f -v
```

The `-f` flag keeps it in the foreground; `-v` increases verbosity. Watch for any errors about missing files, unknown elements, or USB connection failures.

Once this is running, you can interact with Batocera to see what profile LEDSpicer is looking to load, and use this information to decide what profiles you want to create. Creating one for each rom you have may be too much; in that case, create reasonable profiles at the system level (e.g. `arcade.xml` for MAME, FBNeo, Daphne, etcetera) and specific ones for roms that you care more about (e.g `arcade/sf2.xml` for Street Fighter 2).

Step 4.2.1: See what profiles are being searched for

Example output when starting the Galaga rom on the FBNeo system (or any MAME varient):

```
Attempting to load profile: arcade/galaga
Profile cache miss
Reading /userdata/system/configs/ledspicer/profiles/arcade/galaga.xml
Profile failed: Unable to read the file
/userdata/system/configs/ledspicer/profiles/arcade/galaga.xml
Error=XML_ERROR_FILE_NOT_FOUND ErrorID=3 (0x3) Line number=0:
filename=/userdata/system/configs/ledspicer/profiles/arcade/galaga.xml at
line 0
Attempting to load profile: P2_B1
Profile cache miss
Reading /userdata/system/configs/ledspicer/profiles/P2_B1.xml
Profile failed: Unable to read the file
/userdata/system/configs/ledspicer/profiles/P2_B1.xml
```

```
Error=XML_ERROR_FILE_NOT_FOUND ErrorID=3 (0x3) Line number=0:
filename=/userdata/system/configs/ledspicer/profiles/P2_B1.xml at line 0
Attempting to load profile: JOYSTICK2_B1
Profile cache miss
Reading /userdata/system/configs/ledspicer/profiles/JOYSTICK2_B1.xml
Profile failed: Unable to read the file
/userdata/system/configs/ledspicer/profiles/JOYSTICK2_B1.xml
Error=XML_ERROR_FILE_NOT_FOUND ErrorID=3 (0x3) Line number=0:
filename=/userdata/system/configs/ledspicer/profiles/JOYSTICK2_B1.xml at
line 0
Attempting to load profile: arcade
Profile cache miss
Reading /userdata/system/configs/ledspicer/profiles/arcade.xml
Profile failed: Unable to read the file
/userdata/system/configs/ledspicer/profiles/arcade.xml
Error=XML_ERROR_FILE_NOT_FOUND ErrorID=3 (0x3) Line number=0:
filename=/userdata/system/configs/ledspicer/profiles/arcade.xml at line 0
All requested profiles failed
```

In this case, I can see that four different profiles were searched for: `arcade/galaga.xml`, `P2_B1.xml`, `JOYSTICK2_B1.xml`, and `arcade.xm`. As none were found the `default` profile is used.

I may want to create the most generic `arcade.xml` profile to light all LED's with standard colors that map to actions for those buttons, so that any arcade game has a reasonable color map applied. Next I may want to create the slightly more specific `JOYSTICK2_B1.xml` profile to only light the Joysticks (if they are LED) as will as the one active button for each player.

Step 4.2.1: Manually select a profile

To manually trigger a profile load for testing:

```
emitter  LoadProfileByEmulator galaga arcade
```

or:

```
emitter LoadProfile default
```

Step 4.3: Reboot and Verify

Reboot Batocera. Navigate to a game in EmulationStation and observe whether the LEDs change on game selection and launch. If they do not, check:

- `/tmp/ledspicer.log` for daemon errors
- That the scripts are executable
- That `boardId="1"` matches if you have only one board
- That element names in profiles match exactly what is declared in `ledspicer.conf`

5. Tips and Common Pitfalls

Issue	Likely Cause	Fix
Daemon fails to start	Wrong device name in `ledspicer.conf`	Must be exactly `UltimarcUltimate`
LEDs don't change on game launch	Scripts not executable	Run `chmod +x` on desired scripts
Wrong LEDs light up	Pin numbers don't match wiring	Verify harness against board documentation
No USB device found	Driver not loaded or board not detected	Run `lsusb` and check USB connection
Profiles not loading	Name mismatch between profile file and ROM	Check exact ROM filename vs. profile filename
MAME plugin not responding	MAME output not enabled	Add `output = network` to `mame.ini`

6. Conclusion

LEDSpicer fills a gap in the Linux arcade ecosystem by providing a unified, hardware-agnostic daemon for LED management. Its XML configuration model, while requiring some initial effort to map physical pins to named elements, provides substantial flexibility once established. There are many files that each need to be correct for the overall application to work as you want it to.

Reach out to the Discord channel for additional help.

From:
https://wiki.batocera.org/ - **Batocera.linux - Wiki**

Permanent link:
https://wiki.batocera.org/ledspicer

Last update: **2026/06/12 19:57**