xmas-lights-v1

Node Javascript application for controlling Christmas lights using a Raspberry Pi; working monolithic codeset with basic HTTP server for status.

What follows are just some breadcrumbs, as this is easy to forget since it only comse out of storage 1X/year.

Suggest that a script is created called "runlights.sh" and added to the /home/pi/code dirctory. Contents of script:

#!/bin/bash while true; do node /home/pi/code/xmas-lights-v1/xmaslights.js > /home/pi/code/xmas-lights-v1/lights.log 2>&1 && break; done

This script starts the xmaslights.js program using node; it restarts it should the program crash and it directs any terminal output to the "lights.log" file. Invoke this script at boot up by creating the following cron job. Open cron editor with

$ sudo crontab -e

and add the following to the file:

@reboot sudo /home/pi/code/runlights.sh

This will run the "runlights.sh" script at startup.

Should you need to terminate the program, log into the Pi through a terminal and type

$ sudo top | grep node

to discover the PID#, then enter

$ sudo kill -2 PID#

substituting the PID#.

To restart the process w/out a reboot, enter the following from w/in the /home/pi/code directory as root.

$ nohup ./runlights.sh > /home/pi/code/xmas-lights-v1/lights.log

This redirects terminal output to the lights.log file and ignores any "hangups" due to a terminal closing or logged in system suspending.

Project Overview

As the sole developer of LightRunner, I have evolved the project through various iterations. Each generation has employed unique technologies and methodologies, showcasing a journey through hardware and software engineering.

Generations of LightRunner

- **2020's Design (Current Generation)**:
  - **Platform**: Linux running on a 32-bit ARM-based Raspberry Pi.
  - **Technologies**: Node.js, JavaScript.
  - **Hardware**: Eight solid-state relays for controlling light circuits.
  - **Focus**: This repository is dedicated to the current design and its comprehensive implementation.
- **1990's Design**:
  - **Platform**: Motorola M6800 series 8-bit MCU.
  - **Programming**: Assembly language.
  - **Hardware**: Eight TRIACs for light switching, with opto-isolation/coupling to ensure safe interfacing between the TRIACs and the processor.
- **Early Design**:
  - **Platform**: Expansion port on a TRS-80 Color Computer 2.
  - **Programming**: Assembly language.
  - **Hardware**: Utilized TRIACs for light switching, featuring opto-isolation/coupling for processor safety and reliability.