Hudson Build Status Lava Lamps

This Ruby program lights lava lamps based on the status of a Hudson RSS feed— if the most recent of any builds fails, the red lamp will light; if the most recent of each builds succeeds, the program will light the green lamp.

This solution is based on the Pragmatic Programmer's Lava Lamp solution for Cruise Control: http://www.pragmaticautomation.com/cgi-bin/pragauto.cgi/Monitor/Devices/BubbleBubbleBuildsInTrouble.rdoc

You will need:

- Firecracker home automation starter kit: http://www.x10.com/products/firecracker.htm
- 2 differently colored Lava Lamps
- X10 control program bottlerocket http://www.linuxha.com/bottlerocket/
- The Ruby language
- rubygems installed
- the atom gem (to read the RSS feed)

Once you can run the program, it's no trouble to automate this through cron or Hudson.

```ruby
#!/usr/bin/env ruby
require 'net/http'
require 'uri'
require 'rubygems'
require 'atom'

# Light lava lamps based on Hudson build status RSS feed
# Green lamp lights if all builds are good;
# Red lamp lights if any build fails.
#
# Requires atom gem and bottlerocket X10 control software to be installed.
#
class HudsonLava
  def initialize
    @BOTTLE_ROCKET_COMMAND=’/usr/local/bin/br’
    @GREEN_LAMP=’1’ # codes set on X10 lamp modules
    @RED_LAMP=’2’
  end

  def usage
    puts "hudsonLava [hudsonRssFeedUrl]"
    puts " Light lava lamps based on Hudson build status RSS feed."
    puts " hudsonRssFeedUrl: URL of RSS feed for builds you want to monitor; must contain SUCCESS and FAILURE messages."
  end

  def parse_args
    if ARGV.size == 1 then
      @FEED_URL = ARGV[0]
    else
      usage
      exit
    end

  end

  def get_rss_feed(feedUrl)
    feedXml = Net::HTTP.get(URI::parse(feedUrl))
    Atom::Feed.new(feedXml)
  end

  def parse_rss_feed(rssFeed)
    buildList = []
    rssFeed.entries.each { |entry|
      buildLine = "#{entry.title} #{entry.published.strftime('%Y-%m-%d')}"
      buildName, buildNumber, buildStatus, buildDate = buildLine.split
      buildNumber = buildNumber[1..-1]
      buildStatus = buildStatus[1..-2]
      buildList += [[buildName, buildNumber, buildStatus, buildDate]]
    }

end
```
buildDataHash = {}
buildList.each { |buildData|
    buildName, buildNumber, buildStatus, buildDate = buildData

    if buildDataHash.member?(buildName)
        buildDataHash[buildName].addInfo(["#{buildNumber} #{buildStatus} #{buildDate}"])  
    else
        buildDataHash[buildName] = BuildData.new
        buildDataHash[buildName].setName("#{buildName}")
        buildDataHash[buildName].setInfo(["#{buildNumber} #{buildStatus} #{buildDate}"])  
    end
end

buildDataHash
end

def get_success_indicator(buildDataHash)
    success = true
    buildDataHash.each { |buildName, buildData|
        if buildData.getInfo().first =~ /FAILED/      
            puts "#{buildName} failed."
            success = false
        end
    end
    success
end

def lava_lamp(onOff, lampName)
    system "#{@BOTTLE_ROCKET_COMMAND} --#{onOff}=#{lampName}"  
end

def light_green_lamp
    lava_lamp("on", @GREEN_LAMP)
    lava_lamp("off", @RED_LAMP)
end

def light_red_lamp
    lava_lamp("on", @RED_LAMP)
    lava_lamp("off", @GREEN_LAMP)
end

def light_lava_lamp(status)
    if status == true
        puts "No failed builds- lamp is green."
        light_green_lamp
    else
        puts "At least one build failed- lamp is red."
        light_red_lamp
    end
end

def main
    parse_args
    rssFeed = get_rss_feed(@FEED_URL)
    buildDataHash = parse_rss_feed(rssFeed)
    status = get_success_indicator(buildDataHash)
    light_lava_lamp(status)
end

class BuildData
    def initialize
        @buildName = ""
        @buildInfo = []
    end

    def setName(name)
        @name = name
    end
end
def setInfo(info)
    @info = info
end

def addInfo(info)
    @info += info
end

def getInfo()
    return @info
end

def getName()
    return @name
end

end

if $0 == __FILE__
    HudsonLava.new.main
end