ELIXIRCONF™ 2017

Nerves Training Part 5: Phoenix and Kiosks

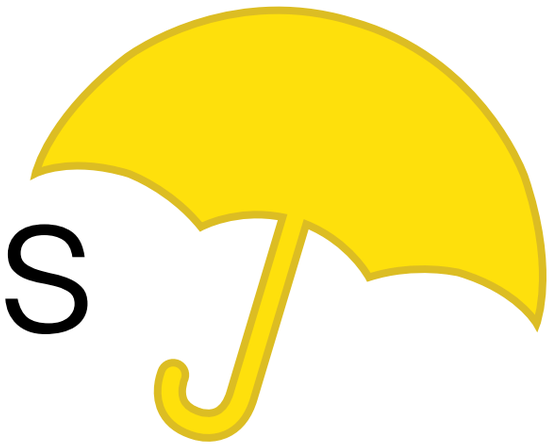
Goals

- Integrate Phoenix into a Nerves application
- Setup and run kiosk project with a web interface

Multi-application projects

- Problem
  - Each project can only produce one OTP application
  - Mammoth OTP applications are possible, but not ideal
- Solutions
  - Umbrellas
  - Ponchos

Umbrella projects



- Mix-supported multi-OTP application projects
- Conveniences for handling configuration including a top-level config file
- Complications when used with Nerves projects
  - Can't build from the base directory
  - Host vs. target confusion

Ponchos

- Plain-old Elixir projects lined up side by side
- Path dependencies between projects
- No mix “conveniences”
- “We call them poncho projects because they protect you from things leaking in from the sides rather than just from above.”
- http://embedded-elixir.com/post/2017-05-19-poncho-projects/

Create the top level project and Phoenix app

```
$ cd <workspace>
$ mkdir webui
$ cd webui
$ mix phx.new phx_app --no-brunch --no-ecto
$ cd phx_app
$ mix phx.server

Check that http://localhost:4000/ works
```

Skip typing

```
$ git clone \
https://bitbucket.org/fhunleth/nervestraining-webui.git webui
$ cd webui
$ git checkout step1
```

Create a RPi Zero App

```
$ cd webui
$ mix nerves.new fw
$ cd fw

Add nerves_init_gadget to mix.exs and config.exs
Copy ssh key config to config/config.exs
```

Skip typing

```
$ git checkout step2
```

Pull in the Phoenix App

```
def deps do
  [{:nerves, "~> 0.7", runtime: false},
   {:phx_app, path: "../phx_app"}] ++
  deps(@target)
end
```

Skip typing

```
$ git checkout step3
```

Import and override Phoenix config

config/config.ex

```
import_config "../../phx_app/config/config.exs"

config :phx_app, PhxAppWeb.Endpoint,
  code_reloader: false,
  server: true
```

Skip typing

```
$ git checkout step3
```

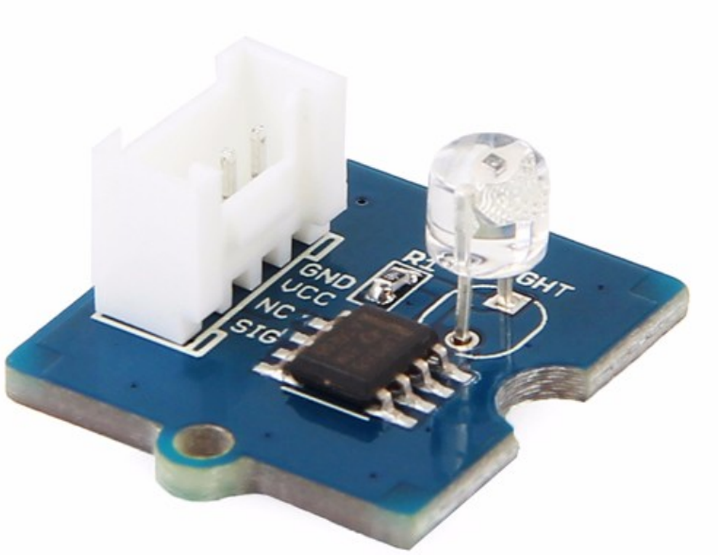
Try it out

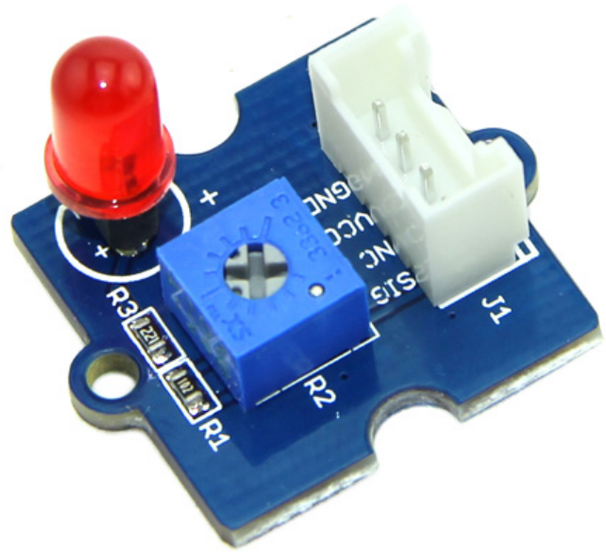
```
$ mix firmware

$ mix firmware.push nerves.local
   -or-
$ ./upload.sh

Open http://nerves.local:4000 up in your browser
```

Connect Phoenix to Hardware





Verify that the light sensor is connected to Port A1 and the LED is connected to Port D3.

Add grovepi to mix deps

```
def deps(target) do
  [
    {:bootloader, "~> 0.1"},
    {:nerves_runtime, "~> 0.4"},
    {:nerves_init_gadget, "~> 0.2"},
    {:grovepi, "~> 0.3"}
  ] ++ system(target)
end
```

Skip typing

```
$ git checkout step4
```

phx_app/lib/phx_app_web/controllers/page_controller.ex

```elixir
defmodule PhxAppWeb.PageController do
  use PhxAppWeb, :controller

  def index(conn, params) do
    case Map.get(params, "led") do
      "on" -> GrovePi.Digital.write(3, 1)
      "off" -> GrovePi.Digital.write(3, 0)
      _ -> :ok
    end

    sensor = GrovePi.Analog.read(1)
    render conn, "index.html", sensor: sensor
  end
end
```

Skip typing

```
$ git checkout step4
```

phx_app/lib/phx_app_web/templates/page/index.html.eex

```
<h1>Light sensor reads <%= @sensor %></h1>

<a href="/?led=on">Turn the LED on!</a>
<p>
<a href="/?led=off">Turn the LED off!</a>
<p>
<a href="/">Refresh</a>
```

Skip typing

```
$ git checkout step4
```

Try it out

```
$ mix firmware

$ mix firmware.push nerves.local
   -or-
$ ./upload.sh

Open http://nerves.local:4000 up in your browser
```

Skip typing

```
$ git checkout step4
```

Nerves and Databases

- PostgreSQL, MariaDB, etc. not normally used on embedded devices

- sqlite - https://github.com/scouten/sqlite_ecto2

- Small flat files - https://github.com/cellulose/persistent_storage

- system_registry - https://github.com/nerves-project/system_registry

Can it recovery from abrupt power failures?
What's the scope of corruption - everything vs. a couple keys?

system_registry

- A hierarchical key-value store with rate-limited notifications
- Uses Elixir's Registry behind the scenes
- Holds configuration and state information for hardware devices
- Examples
  - Notify me when the IP address changes
  - Notify me when hardware is inserted / removed
  - I want this Ethernet interface to be configured using DHCP

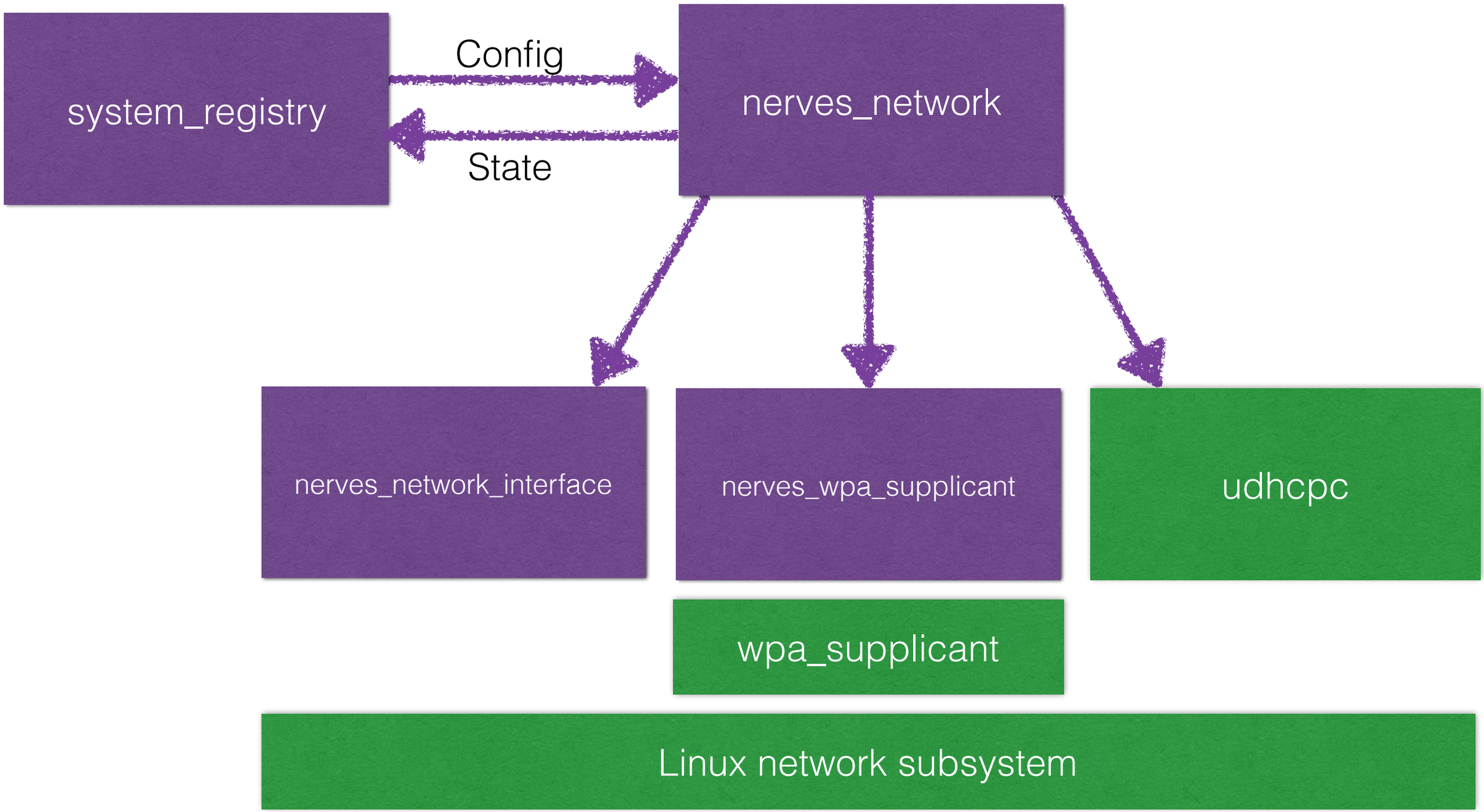
Why system_registry?

- Every Nerves project was creating its own key/value store and change event strategy
- Common bug to get out of sync with events especially when GenServers restart
- Remove need for rate limiters at each point events were being sent of a bandwidth limited channel
- More information at the Thursday keynote!

WiFi

- Raspberry Pi Zero W supports WiFi out of the box!
- nerves_network
  - Pulls network configuration from system_registry
  - Publishes status back to it
  - Nerves.Network.setup helper function

WiFi Components



nerves_network

Network configuration has a lifetime that is linked to the process supplying it.

Try it out

*Important

```
iex> :inet.gethostname
{:ok, 'nerves-00a5'}
iex> Nerves.Network.setup "wlan0", ssid: "nerves1", key_mgmt: :"WPA-PSK", psk:
"nervestraining"
Connect your laptop to the "nerves1" access point
Open http://nerves-00a5:4000 up in your browser

If not working, check IP address:
iex> Nerves.Network.status "wlan0"

If you want to disconnect:
iex> Nerves.Network.teardown "wlan0"

If you just want to change networks, rerun `Nerves.Network.setup/2`
```

See table tent for access point so that we don't overload one network.

Where did the hostname come from?

https://github.com/nerves-project/nerves_system_rpi0/blob/master/rootfs_overlay/etc/erlinit.config

```
# Assign a unique hostname based on the board id
-d "/usr/bin/boardid -b rpi -n 4"
-n nerves-%.4s
```

All Raspberry Pis (and most boards) have unique IDs

```
iex> cmd("cat /proc/cpuinfo")
...
Hardware        : BCM2708
Revision        : 9000c1
Serial          : 000000007b8400a5
```

nerves_network

It's possible to set defaults in config.exs

```
config :nerves_network, :default,
  wlan0: [
    ssid: "nerves1",
    psk: "nervestraining,
    key_mgmt: :"WPA-PSK"
  ]
```

Nerves.Network.setup/2 will override the defaults

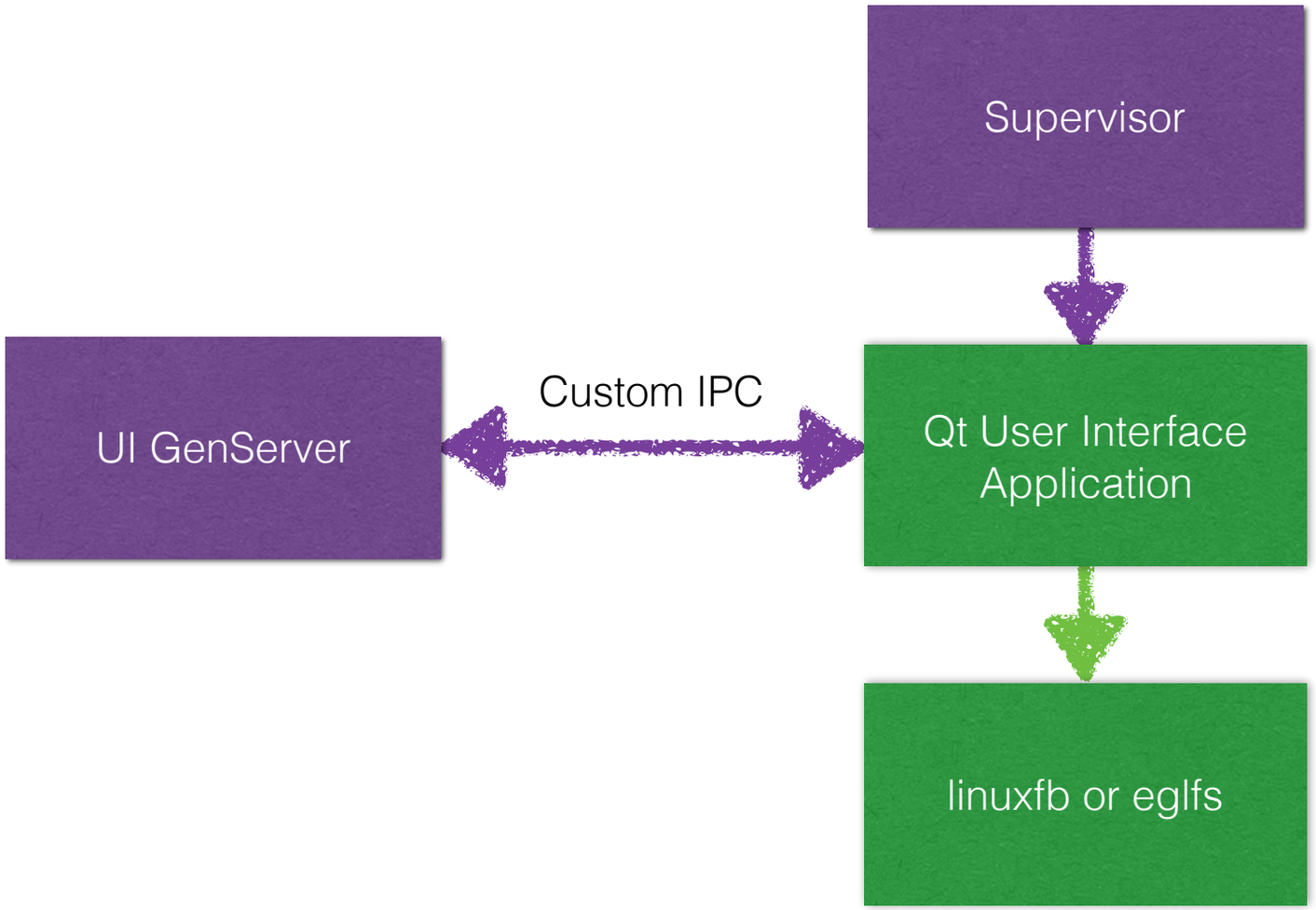
Displays and Kiosks

- Small LCD multi-line displays
  - Built-in fonts; I2C
- Small OLED and LCD displays (320x240 or 640x480)
  - SPI or parallel RGB interface from processor
  - Usually shows up as Linux framebuffer device
- High resolution display panel via HDMI, DVI, LVDS

Embedded UI strategies

- Interactive slideshow
    - No need for graphics libraries - just update framebuffer
- Qt
    - Full featured user interfaces, i18n, multitouch, 3D, QtQuick, etc.
    - Can be made very responsive on low end equipment
- HTML + Javascript
    - Familiar and possible to share code with Web UI components
- No WxWidgets

Integrating Qt



- No full featured Elixir/Erlang binding to Qt (to my knowledge)
- Most UI code is in QtQuick or C++/QWidgets

Integrating web browsers

- Relatively newly supported in Nerves

- Existing systems: Le Tote's `kiosk_system_*` and `nerves_system_qemu_arm`

- Need a reasonably fast target like x86 or RPi3

Web browser options

- Qt WebKit and Qt Chromium options in Buildroot
  - qt-webkit-kiosk is well-used, but deprecated
  - Chromium is massive and BR support is new
- No X11 and Window Managers needed for Qt options
- Input system may need *udev* (select *eudev* in Buildroot to avoid systemd)

Goals

- Target QEMU since it has a emulated display
- Run our Phoenix UI in a local web browser

QEMU

- Open-source machine emulator
  - x86, ARM, SPARC, MIPS, and more
  - Many peripherals emulated as well
- nerves_system_qemu_arm
- Currently SLOWWWWWWWWWWWWWW

Refactor fw/mix.exs

```
def deps(target) do
  system(target) ++
  [
    {:bootloader, "~> 0.1"},
    {:nerves_runtime, "~> 0.4"},
  ]
end
def system("rpi0") do
  [
    {:nerves_system_rpi0, ">= 0.0.0", runtime: false},
    {:nerves_init_gadget, "~> 0.1"},
    {:grovepi, "~> 0.3"}
  ]
end
def system("qemu_arm") do
  [
    {:nerves_system_qemu_arm, ">= 0.0.0", runtime: false},
    {:nerves_network, "~> 0.3"},
    {:nerves_firmware_ssh, "~> 0.1"}
  ]
end
```

Skip typing

```
$ git checkout step5
```

phx_app/lib/phx_app_web/controllers/page_controller.ex

```
def index(conn, _params) do
  #case Map.get(params, "led") do
  #  "on" -> GrovePi.Digital.write(3, 1)
  #  "off" -> GrovePi.Digital.write(3, 0)
  #  _ -> :ok
  #end

  #sensor = GrovePi.Analog.read(1)
  sensor = Enum.random(0..1023)
  render conn, "index.html", sensor: sensor
end
```

Skip typing

```
$ git checkout step5
```

lib/fw/config/config.exs

```
config :bootloader,
  init: [:nerves_runtime, :nerves_network],
  app: :fw

config :nerves_network, :default,
  eth0: [
    ipv4_address_method: :dhcp
  ]
```

Optional, but makes network accessible outside of QEMU

Skip typing

```
$ git checkout step5
```

Build

```
$ cd webui/fw
$ export MIX_TARGET=qemu_arm
$ mix deps.get
$ mix firmware
```

Skip typing

```
$ git checkout step5
```

Create the QEMU start script and starting image

One-time command

```
$ export MIX_TARGET=qemu_arm

$ mix nerves.gen.qemu_script
Check for run-qemu.sh

$ mix firmware.image
Check for fw.img

Start QEMU
$ ./run-qemu.sh
Wait for DHCP messages
```

Skip typing

```
$ git checkout step5
```

Start qt-webkit-kiosk

```
iex> :os.cmd('qt-webkit-kiosk -platform linuxfb
     -c /etc/qt-webkit-kiosk.ini -u http://127.0.0.1:4000')

Wait
```

Skip typing

```
$ git checkout step5
```

Autostart the Browser

lib/fw/application.ex

```
  def start(_type, _args) do
    children = [
      worker(Task, [&start_kiosk/0], restart: :permanent)
    ]

    ...
    Supervisor.start_link(children, opts)
  end

  def start_kiosk() do
    :os.cmd('qt-webkit-kiosk -platform linuxfb' ++
      '-c /etc/qt-webkit-kiosk.ini -u http://127.0.0.1:4000/')
  end
```

Skip typing

```
$ git checkout step6
```

Speaker Schedule: Thu Sep. 7

Time		Speaker
07:30 AM - 09:00 AM		Registration / Breakfast
09:00 AM - 09:10 AM		Welcome
09:10 AM - 10:10 AM		Keynote - Justin Schneck
10:10 AM - 10:40 AM		Morning Break
10:40 AM - 11:20 AM	Grand *Elixir* (BEG-ADV)	**Veronica Lopez** *Alchemist Gopher: My Journey from Go to Elixir*
	Evergreen E *Elixir* (BEG-ADV)	**Boyd Multerer** *Elixir Native UI*
	Evergreen AB *Plug* (INT-ADV)	**Gary Rennie** *HTTP/2 Plug to Phoenix, Cowboy too*

ELIXIRCONF™ 2017

Nerves Training Part 5: Phoenix and Kiosks