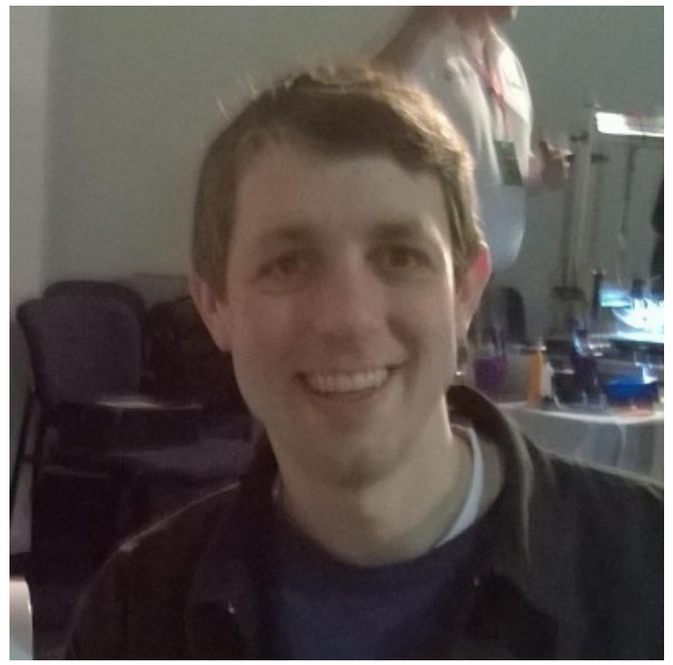
ELIXIRCONF™ 2017

Nerves Training
Part 1: Getting started

Who am I?



- Embedded software developer for most of my career
- Started the Nerves Project in 2013 and one of many on the Nerves Core team
- Run a CoderDojo in the Washington, DC area
- Runner
- @fhunleth on GitHub, Twitter, the Elixir Lang Slack, LinkedIn

Others on the Nerves Core Team

Justin Schneck

mobileoverlord

Garth Hitchens

ghitchens

Greg Mefford

GregMefford

Jeff Smith

electricshaman

Connor Rigby

ConnorRigby

Christopher Coté

entone

Tim Mecklem

tmecklem

Plus many awesome contributors!

Github contributors on August 20, 2017

Nerves in Production

LE TOTE

ROSE POINT
NAVIGATION SYSTEMS

NATIONAL ASSOCIATION of REALTORS®

FarmBot

Training Notes

- Goals
  - Make you comfortable creating embedded software with Nerves
  - Introduce important embedded systems topics
  - Have fun
- Typing example code is technically optional, but please try
- Most of the fun in past training has been had by people extending the examples
- Nerves team volunteers are here to help
- *#nerves_training* channel on the elixir-lang slack

HALP!

- Crosscompilers + hardware + large downloads means things will go wrong

- Flip your table tent if you want to debug while waiting for help

- We have extra hardware to swap if debugging looks time consuming

- If you're ahead, feel free to offer help your table mates - some people find really interesting problems

Training Schedule

Day 1

- Getting started
- Digital and analog I/O
- Digital buses and I2C
- Nerves Systems Part 1

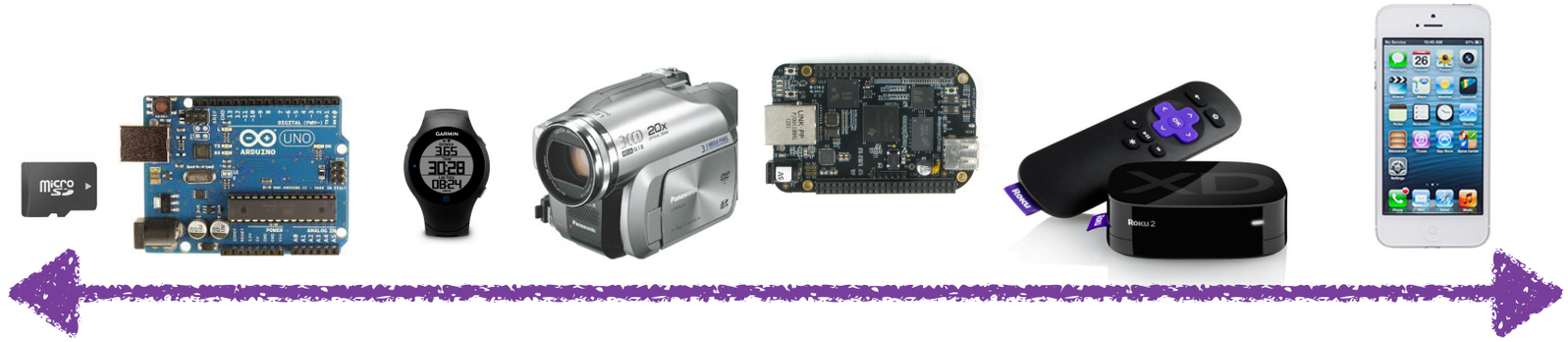
Day 2

- Nerves Systems Part 2
- Integrating Phoenix and creating Kiosk Devices
- Cameras and streaming video
- Advanced Nerves Topics*

Embedded system

A computer system with a dedicated function

Where does Nerves fit?



Limited resource/
Specific purpose

General
purpose

Microcontrollers

RTOS

Embedded Linux

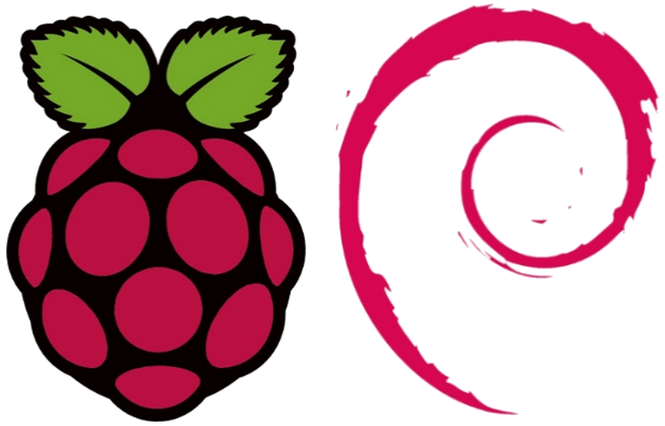
Android, iOS,
Ubuntu, Windows

32-bit Microprocessors

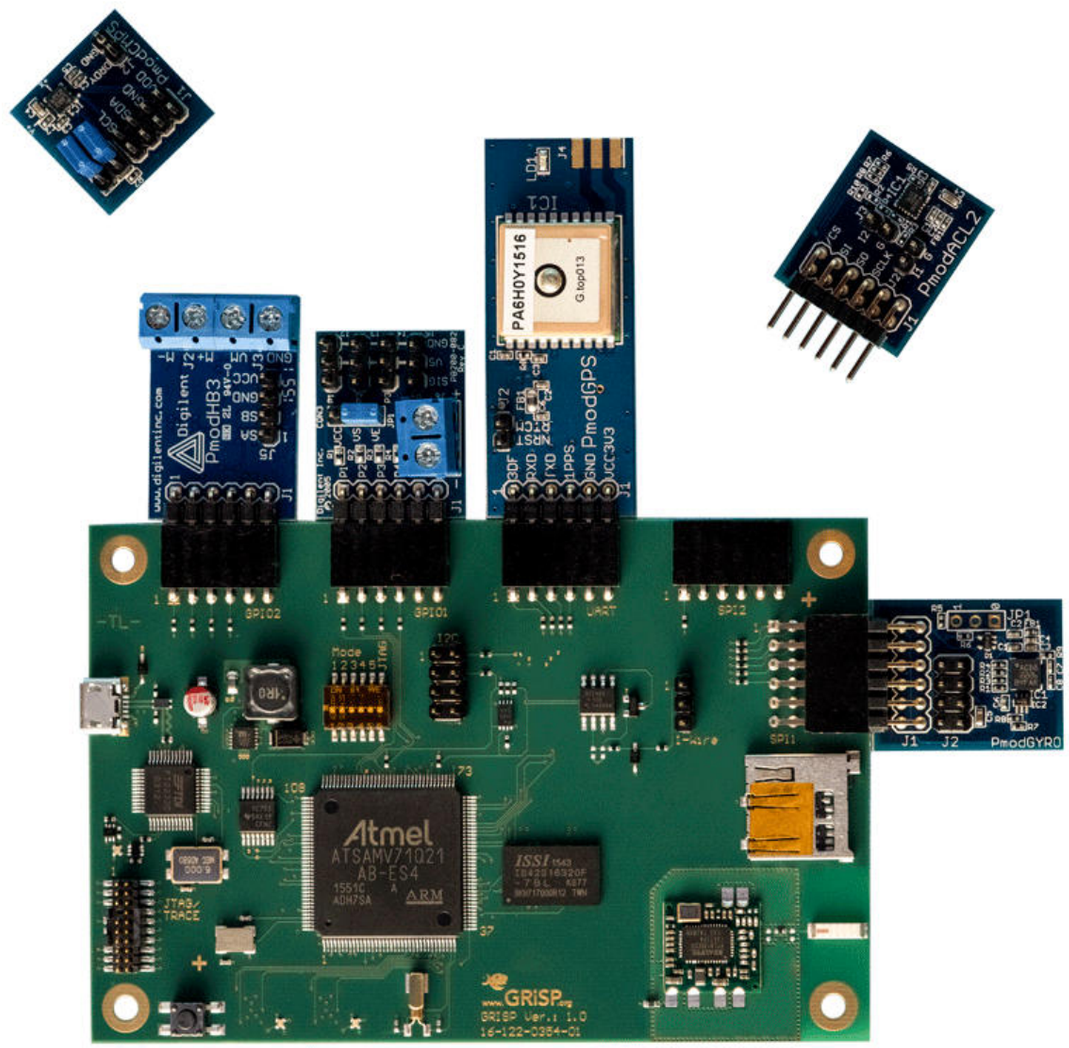
64-bit

- Devices need to be capable of running Linux to support Nerves
- Sweet spot is for network connected devices with built-in smarts

Other embedded Elixir options



Develop on Raspbian
(and then switch to Nerves)



GRiSP - Erlang on RTEMS

Host

The computer on which you are editing source code, compiling, and assembling firmware

Target

The computer that runs the compiled code such as a Raspberry Pi Zero, Beaglebone, or other board

Toolchain

The tools required to build code for the target such as compilers, linkers, binutils, and a C runtime

System

A lean Buildroot-based Linux root filesystem that has been customized and cross-compiled for a particular target

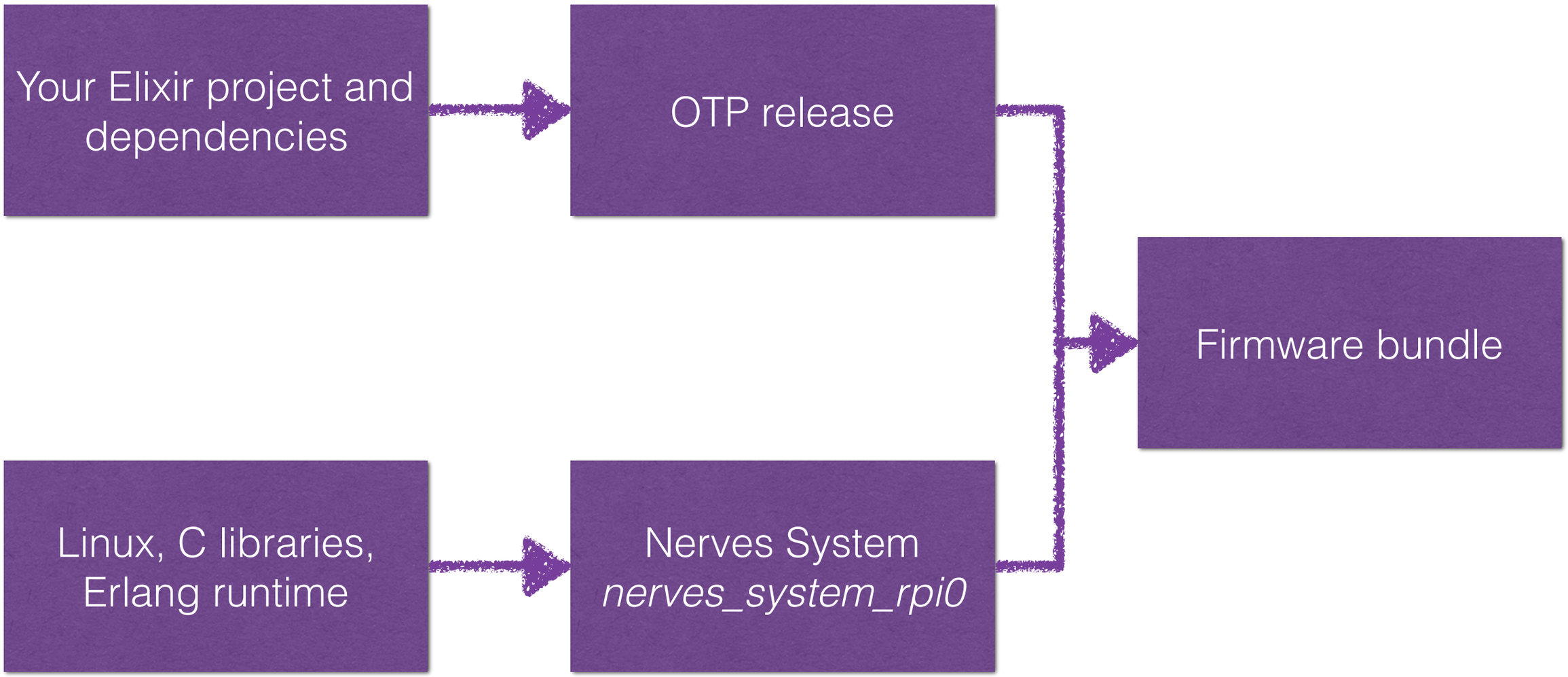
Firmware bundle

A single file that contains the system, application, configuration and anything else needed to initialize and update the non-volatile storage of a device

Firmware image

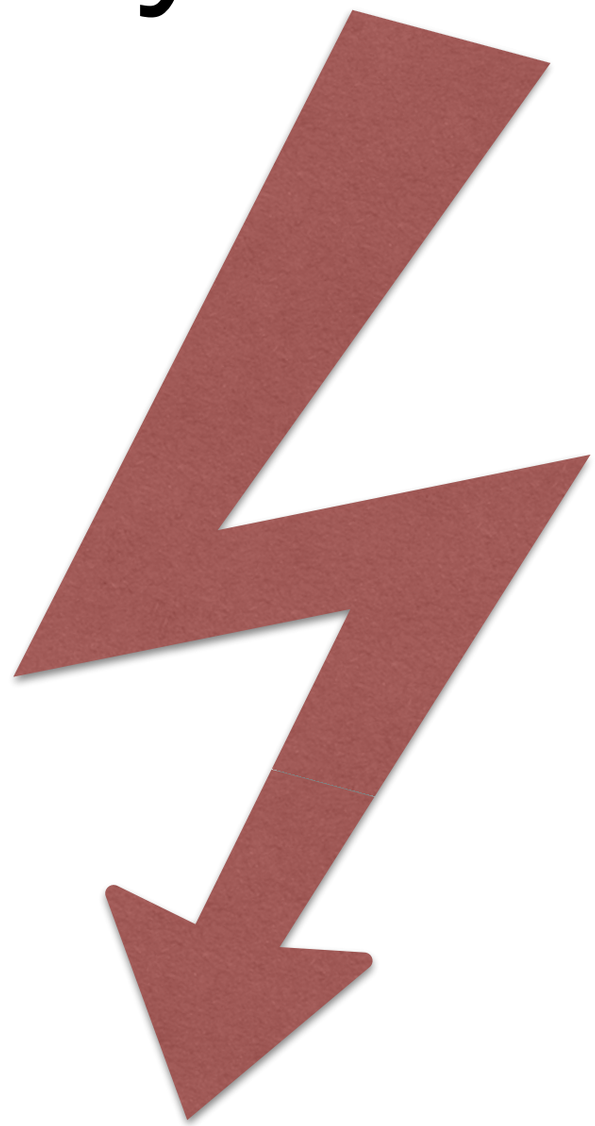
A large binary file generated from the firmware bundle that is a bit-for-bit image of a devices non-volatile memory. It can be used by tools like *dd(1)* and etcher.io.

Nerves

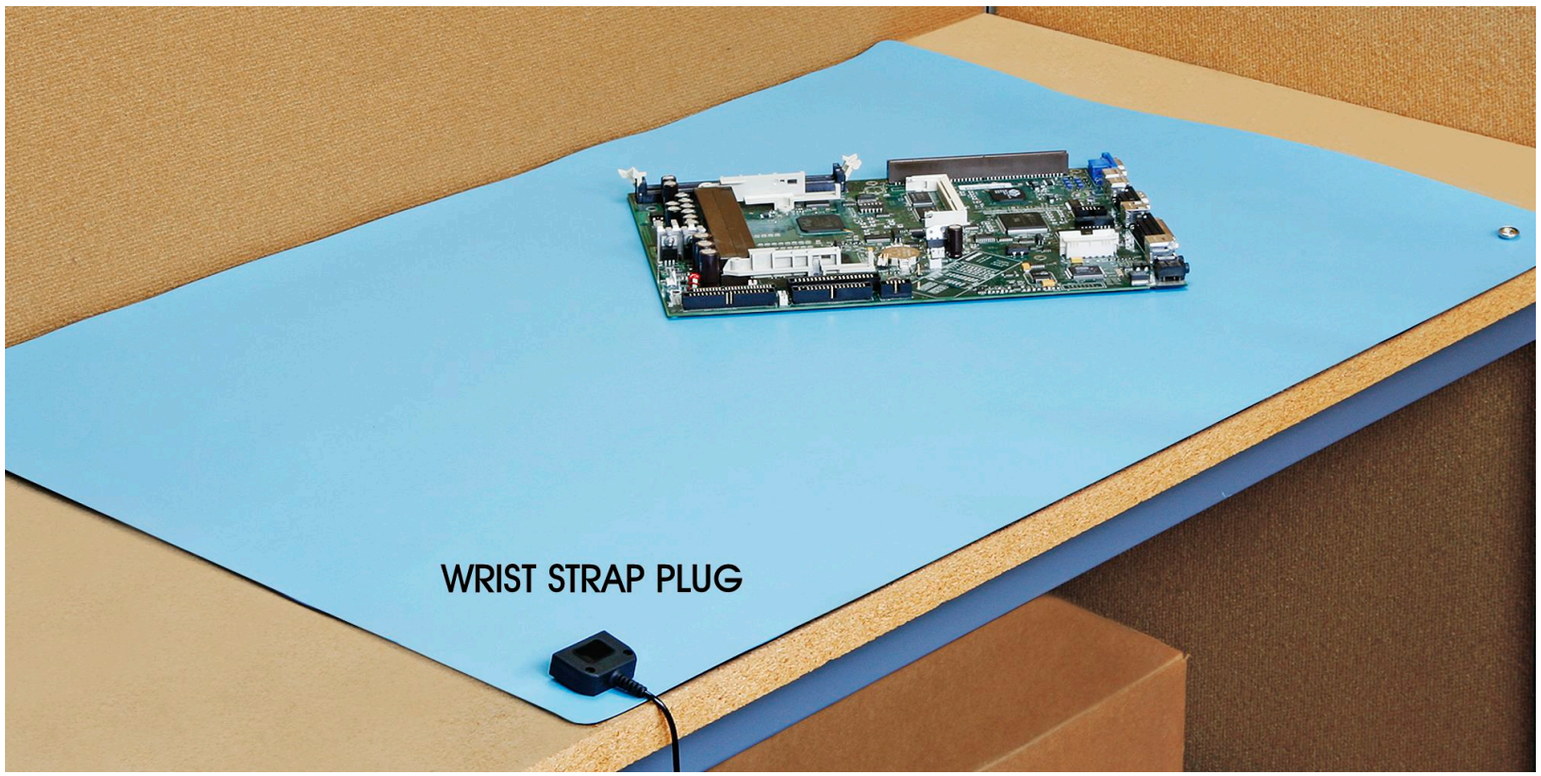

Your Elixir project and dependencies
OTP release
Firmware bundle
Linux, C libraries, Erlang runtime
Nerves System
nerves_system_rpi0

Static electricity

- Static electricity breaks electronics!
- Sometimes after the fact and in weird ways
- Some electronics are more susceptible than others (like MOSFETs)



Normal precautions

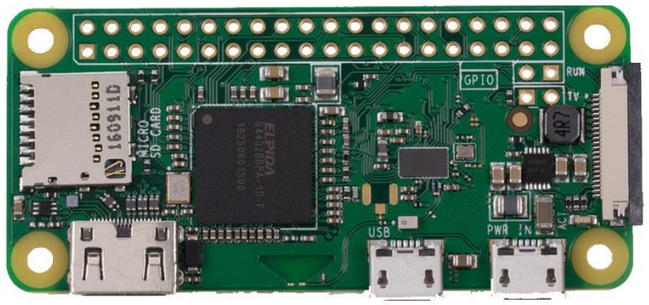


For today

- Try to hold boards by their edges
- Touch grounded parts of your laptop periodically (if it's metal, it's usually grounded)
- Prefer to plug/unplug the USB cable to the Raspberry Pi Zero from your laptop
- Shouldn't need to be said, but don't take your shoes off and rub your socks on the carpet today

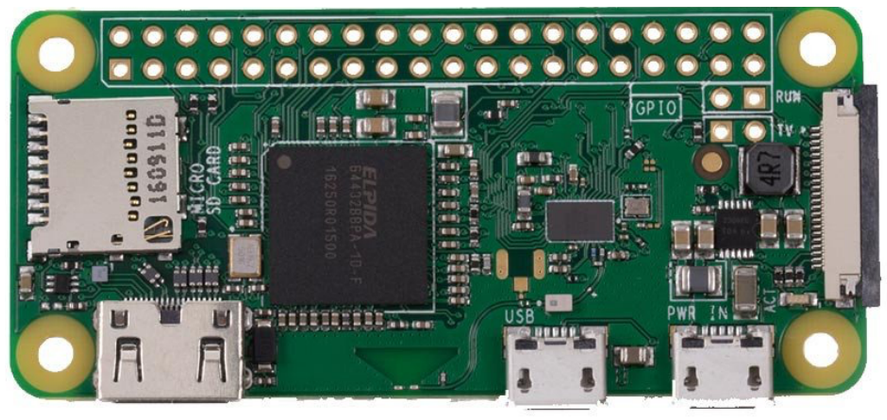
First hardware

- Raspberry Pi Zero W
- USB Cable
- MicroSD card
- Label



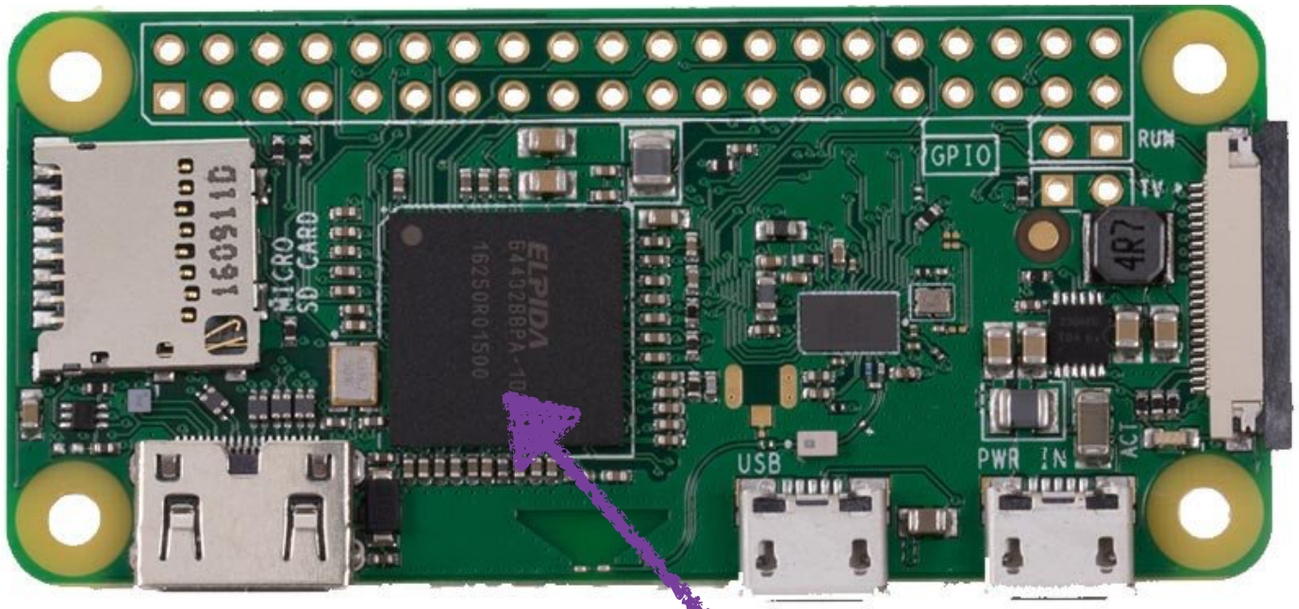
Raspberry Pi Zero W

- 1 GHz, single-core ARM CPU
- 512 MB DRAM
- USB On-The-Go port

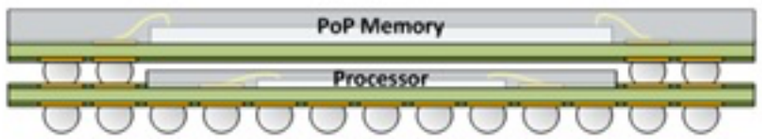


  - Called *host* or *gadget* mode by Linux depending on the direction
  - Not on the Raspberry Pi B+, 2, 3
- 802.11 bgn wireless
- MicroSD slot for code and storage
- Camera connector

Cool things about the RPi0W

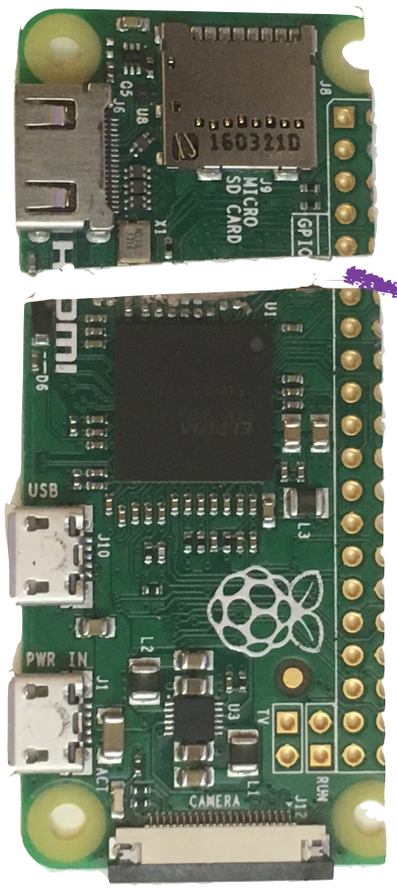


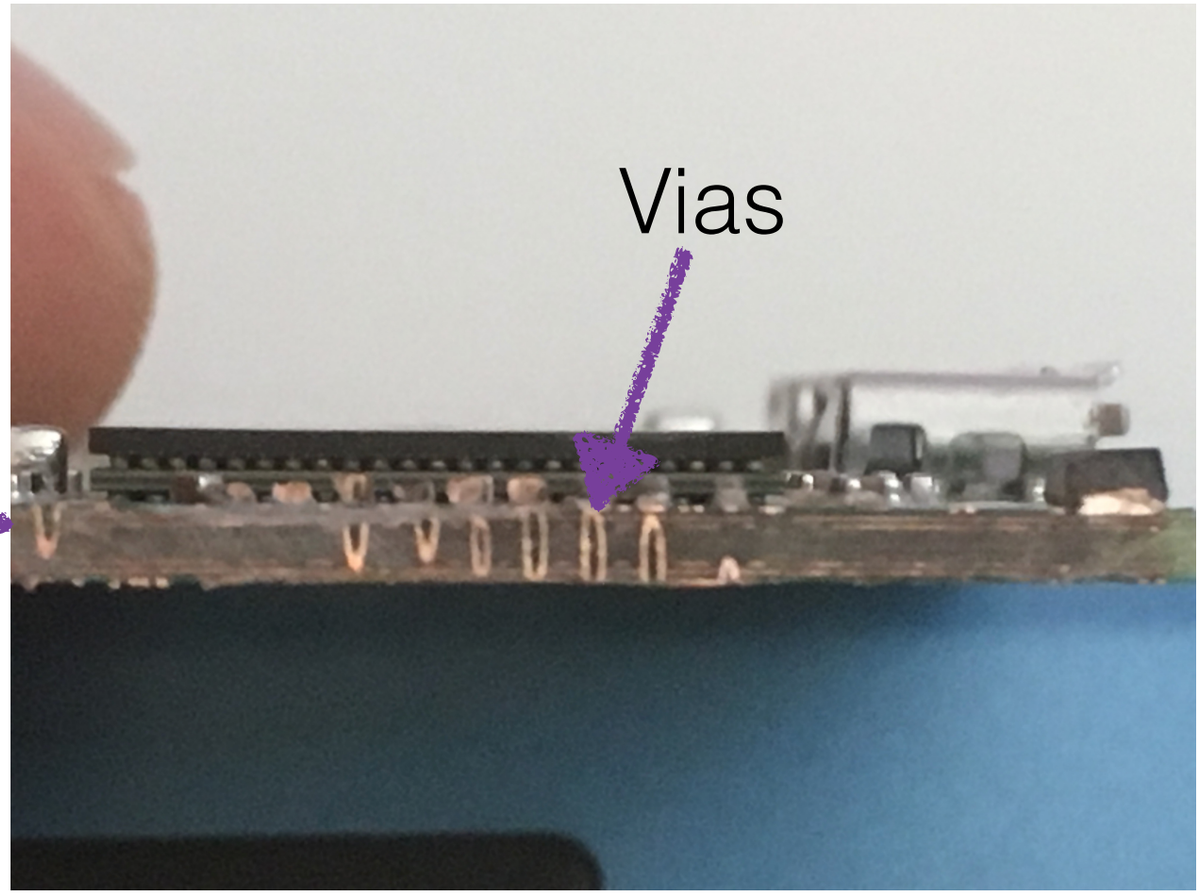
Package on package (PoP) Memory (DRAM)



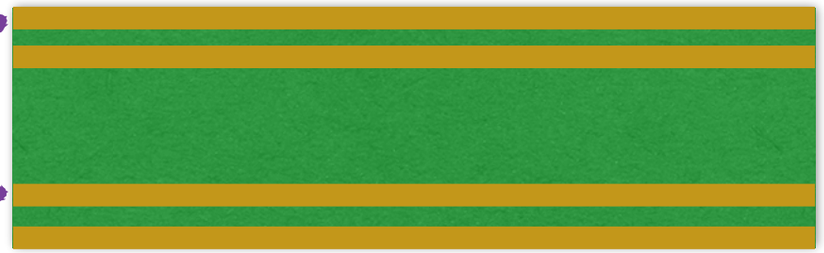
[Image source: http://corelis.com/]

Raspberry Pi Zero PCB

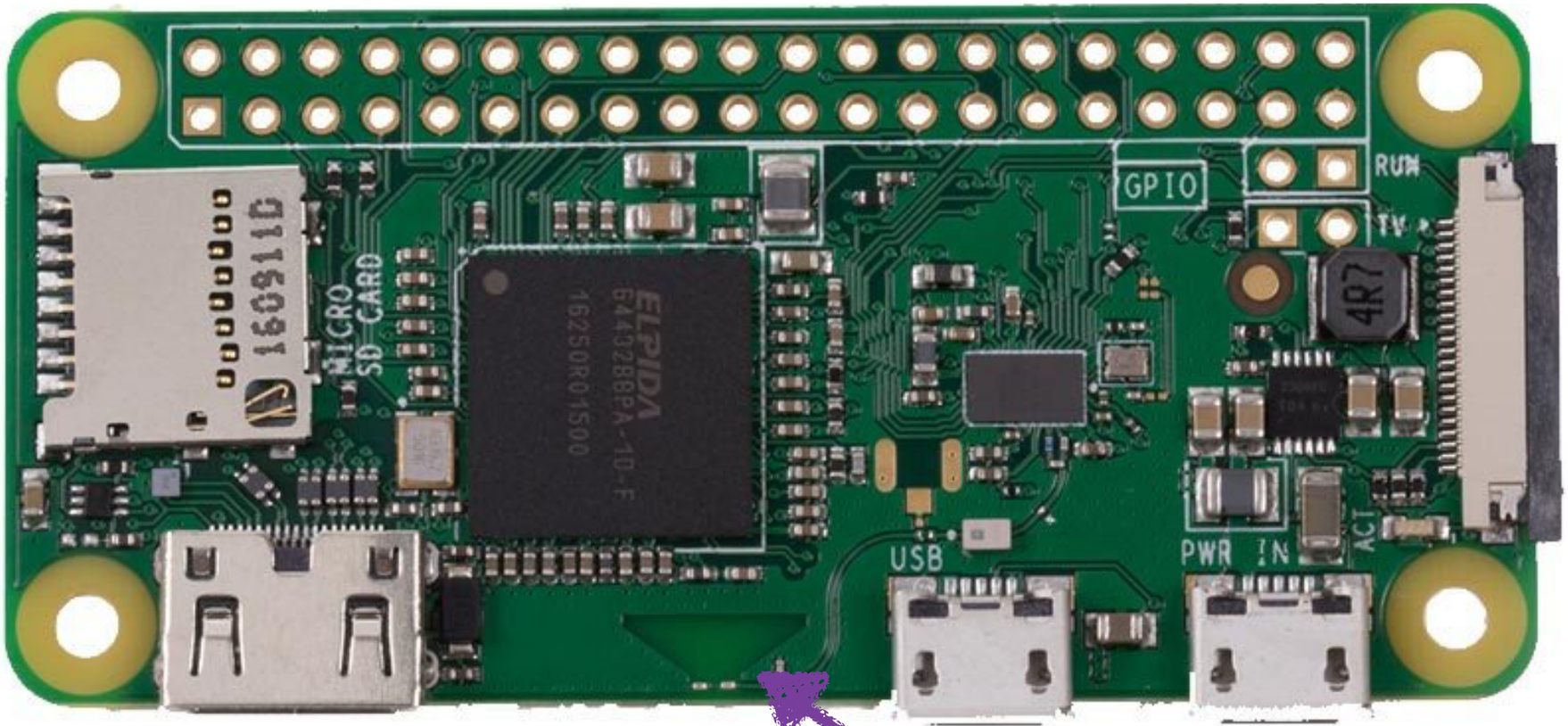




4 Copper layers
Mostly fiberglass in between



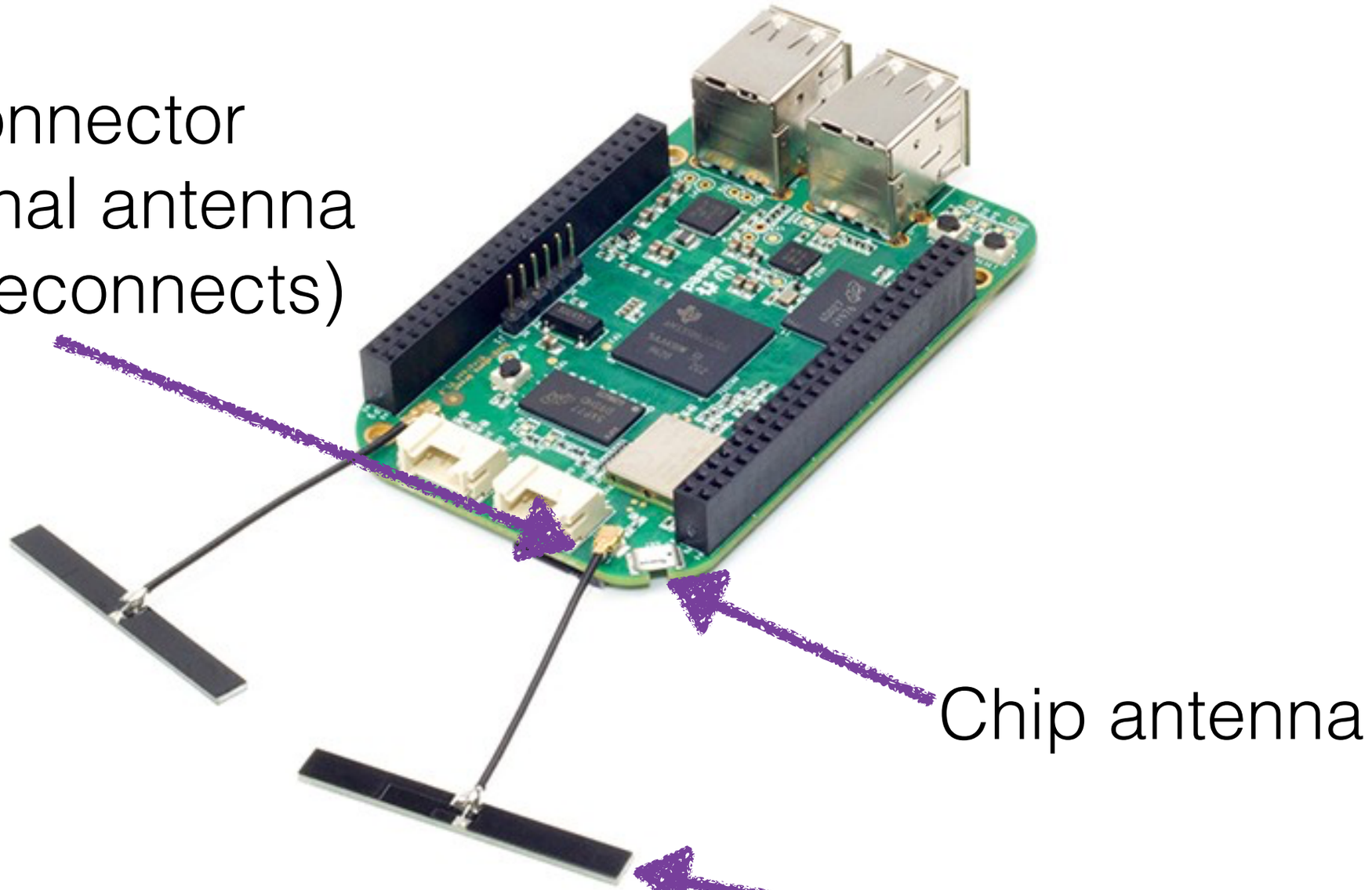
More RPi Zero W Coolness



WiFi resonant cavity antenna

More common antennas

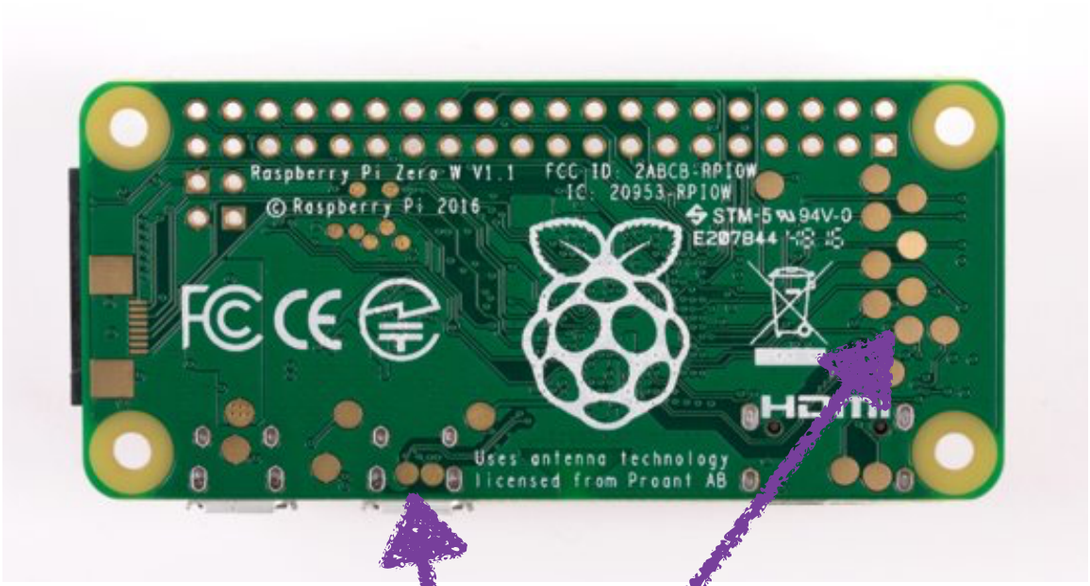
U.FL connector
to an external antenna
(limited # reconnects)



Chip antenna

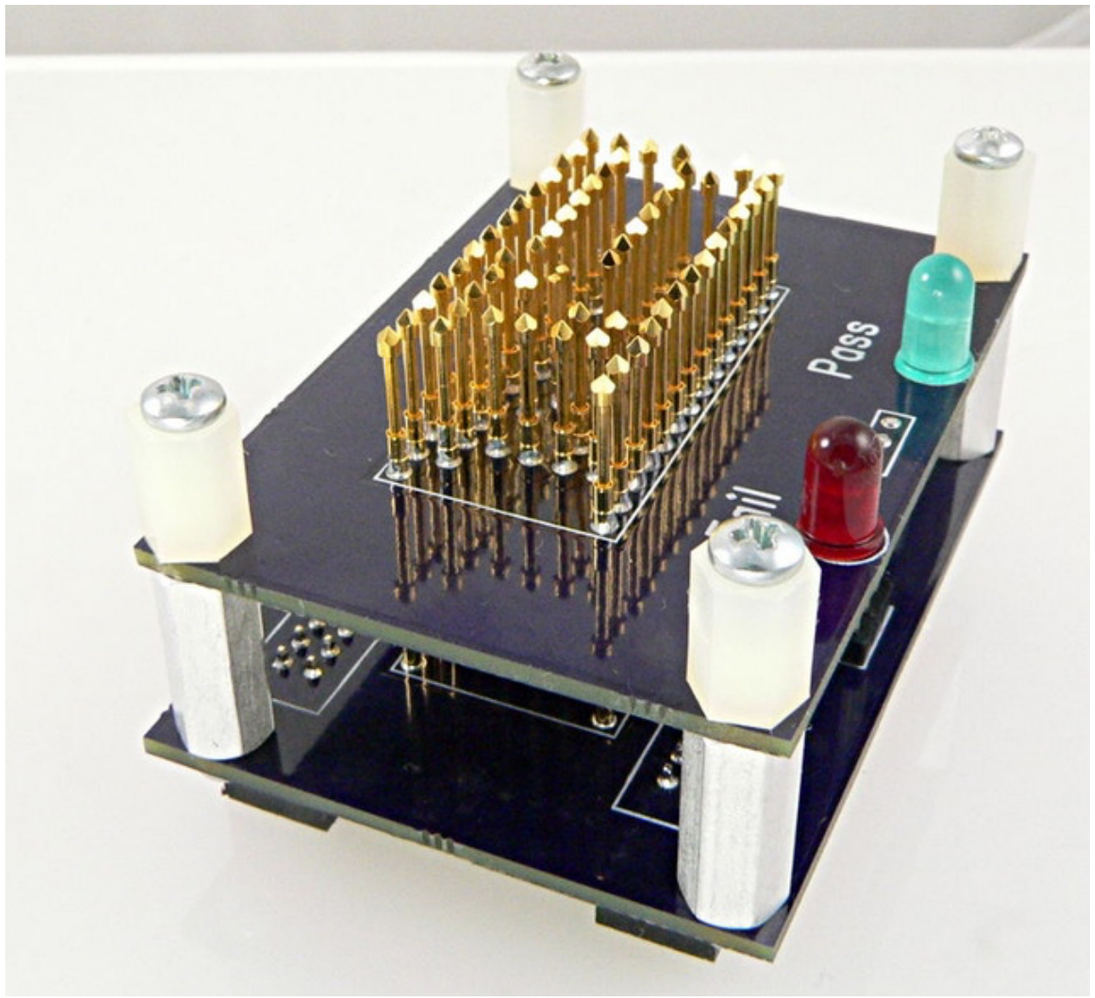
PCB antenna

Testing in manufacturing



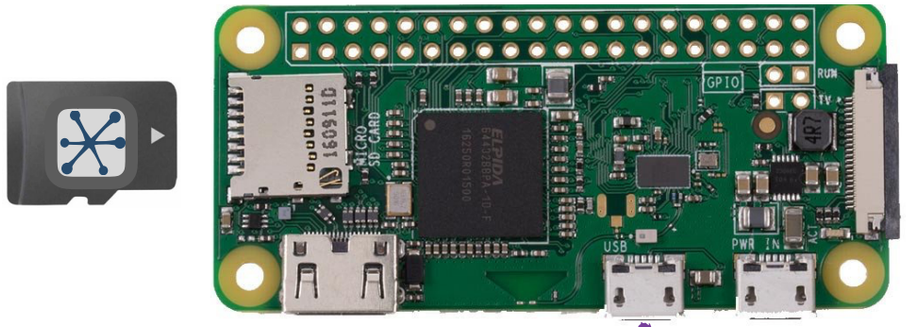
Test points used in manufacturing

Example “bed of nails”
(not the RPi0W one)



https://www.kickstarter.com/projects/paulstoffregen/teensy-30-32-bit-arm-cortex-m4-usable-in-arduino-a/posts/305527

Initial goals

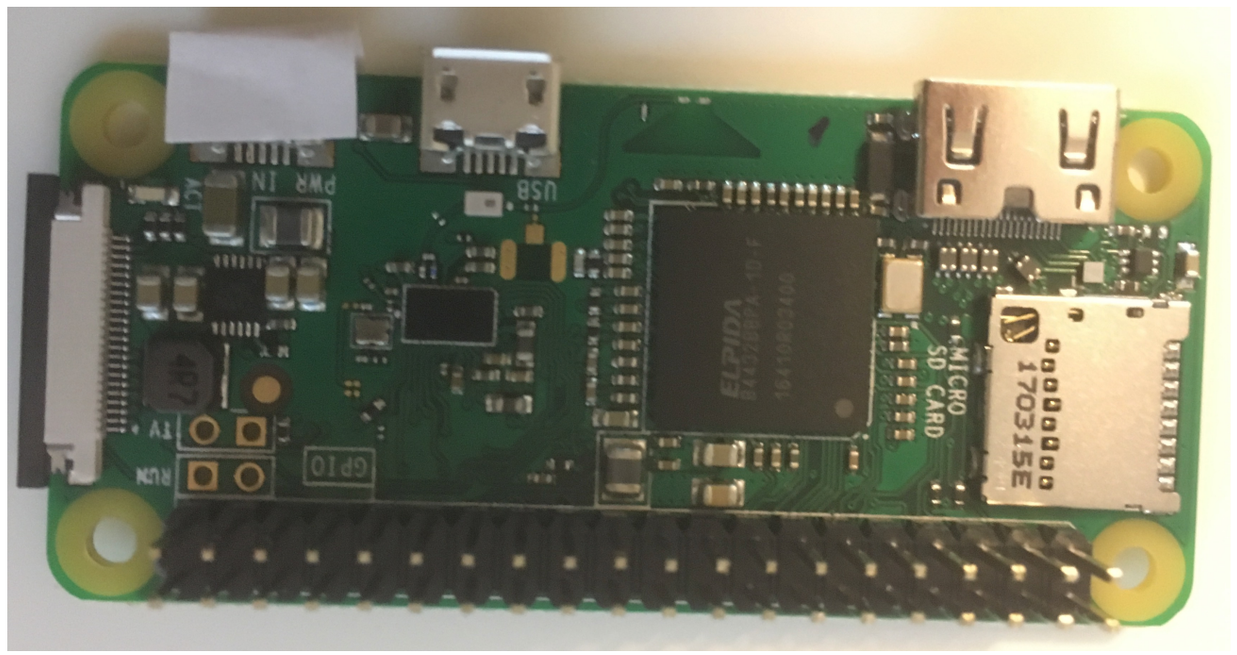


USB w/ virtual serial port
and virtual Ethernet

- Create a trivial Nerves project
- Access the IEx console
- Upgrade the image using ssh

Cover “PWR IN” connector

- Power, console, and Ethernet are all through the connector marked USB
- Connecting to “PWR IN” is close to the #1 confusing HW mistake
- Put sticker over the “PWR IN” connector to not forget
- Feel free to take a second sticker and label the back of your RPI0 with your name



Install Nerves

- https://hexdocs.pm/nerves/installation.html

- Most important:

```
$ mix archive.install https://github.com/nerves-project/archives/raw/master/
                       nerves_bootstrap.ez
```

or

```
mix local.nerves
```

Pre-populate downloads

- Normally don't have to do this - feel free to skip
- Connect to the test WiFi network (nerves1, nerves2, or nerves3)
- Go to ftp://192.168.11.1
- Download the files in the *dl* directory
- Place in *~/.nerves/dl*

Following along with git

- Each step is a branch
- If you're typing, run *git diff step[n+1]* to compare your work for typos
- Customizations?
    - Create your own branches and work in parallel
    - Or *git stash* and *git stash pop* when advancing steps
- Cleaning up?
    - Undo your modifications: *git reset --hard*
    - Erase untracked files and directories: *git clean -fdx*

mix nerves.new

```
$ cd <workspace>
$ mix nerves.new starter
$ cd starter
```

Skip typing

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

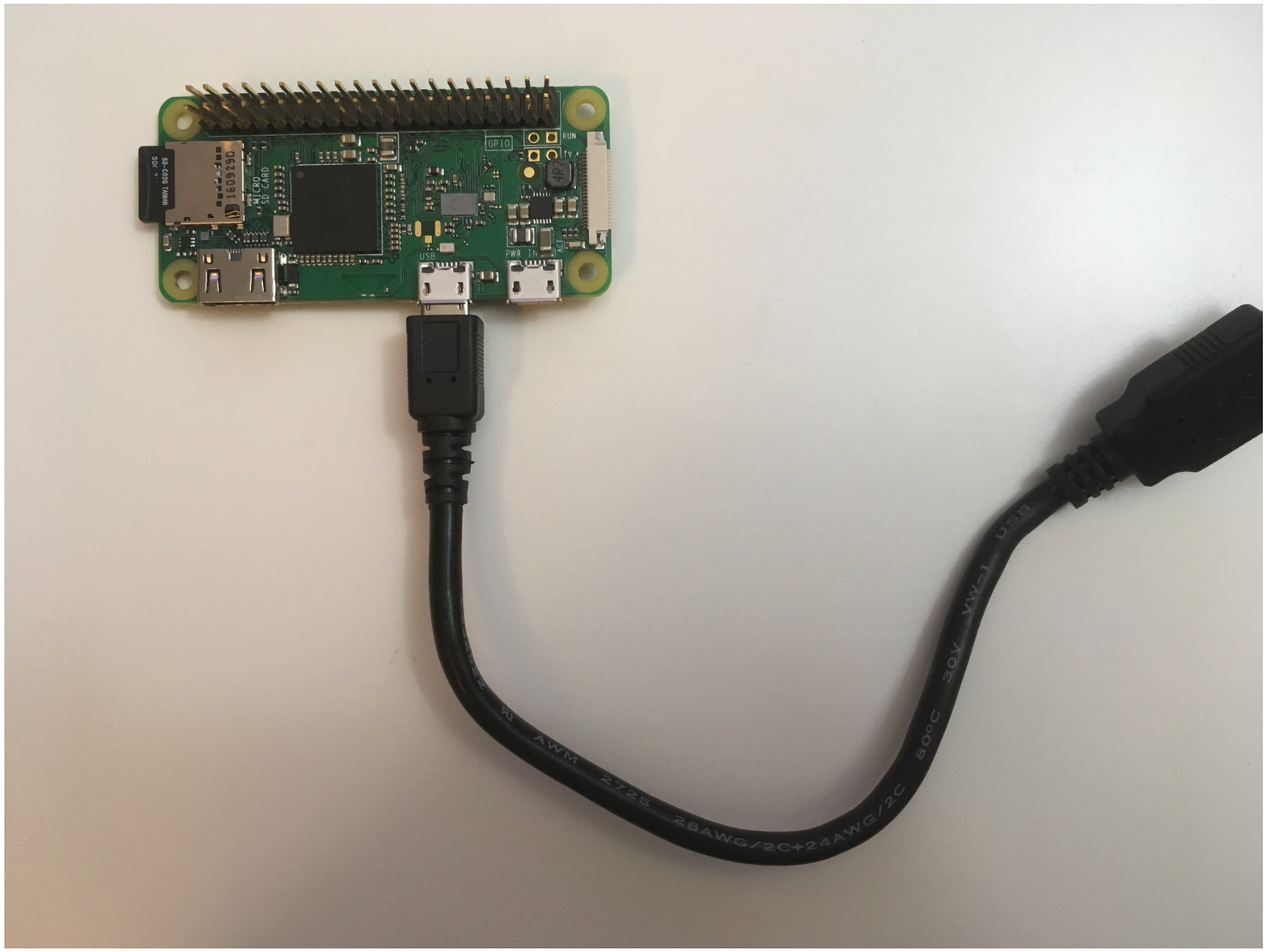
Build and Burn

```
$ export MIX_TARGET=rpi0
$ mix deps.get
$ mix firmware

See that you have a firmware bundle
$ find . -name "*.fw"

Insert a MicroSD card
$ mix firmware.burn
```

Connect up the Zero



Connect to the board

If using screen

```
$ screen /dev/tty.usb<device>
CTRL+a, CTRL+\ to exit
```

If using picocom

```
$ picocom /dev/tty.usb<device>
CTRL+a, CTRL+x to exit
```

nerves_init_gadget

- Sets up a link local network over the virtual USB Ethernet
- Adds mDNS support so that nerves.local works
- Adds SSH push-based firmware updates

mix.exs

```
def deps(target) do
  [ system(target),
    {:bootloader, "~> 0.1"},
    {:nerves_runtime, "~> 0.4"},
    {:nerves_init_gadget, “~> 0.1”}
  ]
end
```

Skip typing

```
$ git checkout step2
```

config/config.exs

```
# Request bootloader to start nerves_init_gadget
config :bootloader,
  init: [:nerves_runtime, :nerves_init_gadget],
  app: :starter
```

Skip typing

```
$ git checkout step2
```

config/config.exs

~/.ssh/id_rsa.pub

```
ssh-rsa AAAAB3N…
```

```
config :nerves_firmware_ssh,
  authorized_keys: [
    "ssh-rsa AAAAB3N…",
    "ssh-rsa another one",
  ]
```

— or —

or if you can't find your id_rsa.pub

```
$ ssh-keygen -y -f ~/.ssh/id_rsa
AAAAB3…
```

```
config :nerves_firmware_ssh,
  authorized_keys: [
    File.read!(Path.join(System.user_home!,
                         ".ssh/id_rsa.pub"))
  ]
```

Skip typing

```
$ git checkout step2
```

Try it out

```
$ export MIX_TARGET=rpi0
$ mix deps.get
$ mix firmware

Insert MicroSD card
$ mix firmware.burn

Plug in Raspberry Pi Zero and wait ~30 seconds
$ ping nerves.local
```

lib/starter.ex

```
# Add a function (could be anything)
def hello() do
  :world
end
```

Skip typing

```
$ git checkout step3
```

Pushing a firmware update

If you don't have a password protecting your private key

```
$ mix firmware.push nerves.local
```

If you have a password protected private key or anything where you prefer regular commandline ssh

```
$ wget https://github.com/fhunleth/nerves_firmware_ssh/raw/master/upload.sh
$ chmod +x upload.sh
$ ./upload.sh
```

Skip typing

```
$ git checkout step3
```

Try it out

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

Wait for device to reboot

$ picocom /dev/tty.usb*

If the upload doesn't work, check the authorized keys on the device by running:
iex> Application.get_all_env(:nerves_firmware_ssh)

iex> Starter.hello
:world

Try making another change and uploading it
so that you become comfortable with this
development cycle.
```

Skip typing

```
$ git checkout step3
```

Working from the IEx prompt

- CTRL+C doesn't exit in Nerves
- Nerves.Runtime.Helpers
  - *Nerves.Runtime.Helpers.install()*
    - *use Nerves.Runtime.Helpers*
    - Extra IEx helpers like *cmd/1, reboot!/0*, and *hex/1*
- Nerves.Runtime.Shell
  - Sort of a bash-like shell
  - CTRL+G, then "s sh", then "c"

bootloader

- Elixir application and Distillery plugin that separates your project's initialization
    - Critical initialization - if it fails, crash
    - Non-critical initialization - if it fails, log it
- Bootloader fixes MicroSD card swap development by letting you put your application in the non-critical section
- https://github.com/nerves-project/bootloader

bootloader appreciation time

lib/starter/application.ex

```
defmodule Starter.Application do
  use Application

  def start(_type, _args) do
    import Supervisor.Spec, warn: false

    children = [
       worker(Starter.Worker, []),
    ]

    opts = [strategy: :one_for_one, name: Starter.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

Skip typing

```
$ git checkout step4
```

Doesn't exist

Try it out

```
$ mix firmware

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

Repeatedly run picocom or screen to catch log messages
for crash
$ picocom /dev/tty.usb*

Observe that you can still ping the board and upload
new firmware
```

Skip typing

```
$ git checkout step4
```

PAIN

rel/config.exs

```
release :starter do
  set version: current_version(:starter)
  #plugin Bootloader.Plugin
  if System.get_env("NERVES_SYSTEM") do
    set dev_mode: false
    set include_src: false
    set include_erts: System.get_env("ERL_LIB_DIR")
    set include_system_libs: System.get_env("ERL_SYSTEM_LIB_DIR")
    set vm_args: "rel/vm.args"
  end
end
```

Comment out the Bootloader plugin to disable it

Skip typing

```
$ git checkout step5
```

Try it out

```
$ mix firmware

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

Repeatedly run picocom or screen to catch log messages
for crash
$ picocom /dev/tty.usb*

Cry

Revert your changes and reprogram the MicroSD card
$ mix firmware
$ mix firmware.burn
```

Skip typing

```
$ git checkout step5
```

Recovering

- Undo the changes
- To recover, you need to remove the MicroSD card and run “mix firmware.burn”
- Some observations
  - The crash is hard to catch on the serial port
  - mix firmware.burn erases everything

FTDI / UART cables

- Key attributes
  - Logic voltage level - 5V or 3V
  - 6 pin header vs. wires
  - 5 V power supply?
  - FTDI, Prolific, or clone chip

- Sold by Adafruit, Digikey, Microcenter, Mouser, Sparkfun, etc.
- Except for the clone chip versions, all work on Linux, Mac, and Windows

Erlang Distribution

- Seamless message passing between Erlang nodes
- Remote shell, observer and other debug tools
- Some hurdles to using with network interfaces that aren't always available
- *nerves_init_gadget* can configure it

config/config.exs

```
config :nerves_init_gadget,
  node_name: "nerves"
```

Once the virtual Ethernet interface comes up, *nerves_init_gadget* will start *epmd* and name the node :"nerves@nerves.local"

Skip typing

```
$ git checkout step6
```

mix.exs

```
def application(_target) do
  [mod: {Starter.Application, []},
   extra_applications: [:logger, :runtime_tools]]
end
```

runtime_tools is needed for Observer and other tracing tools.

Skip typing

```
$ git checkout step6
```

Try it out

```
$ mix firmware

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

Grab the Erlang distribution cookie
$ cat rel/vm.args
Grab the node name (pinging nerves.local works too)
$ picocom /dev/tty.usb*

Remsh to the board
$ iex --name me@0.0.0.0 --cookie <from above> --remsh nerves@nerves.local

Or start up observer
$ iex --name me2@0.0.0.0 --cookie <from above>
iex> :observer.start()
Nodes->Connect node
```

Skip typing

```
$ git checkout step6
```

Going forward in training

- Upgrade your board via *ssh* from now on (no MicroSD card swaps)
- All future projects start with the *nerves_init_gadget* based app we made
- No more Erlang distribution, but feel free to add to use *remsh* and *:observer*
- *Logger.configure level: :info* and *CTRL+L*

The cheatsheet has details on all of this.

ELIXIRCONF™ 2017

Nerves Training
Part 1: Getting started