ELISA
Enabling Linux in
Safety Applications

WORKSHOP

NASA Goddard

Building an OSS Ecosystem for Space

Tim Bird
Sony Electronics

Who am I?

- Principal Software Engineer at Sony Electronics
  - At Sony for over 20 years, including time as Sony's Linux kernel maintainer
- Member of Linux Foundation Board of Directors
- Creator and organizer of Embedded Linux Conference (started in 2005)
- Former CTO of Lineo, an early embedded Linux company
- Working with Linux and OSS for over 30 years
- E-mail: tim.bird@sony.com

Photo by Jake Edge, LWN.ne

Lessons from OSS in space

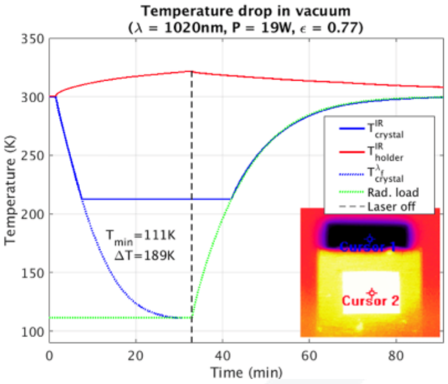
ELISA Enabling Linux in Safety Applications

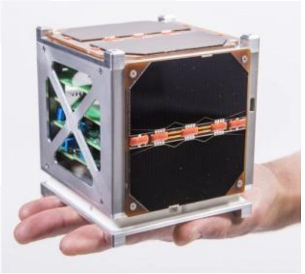
WORKSHOP

SPACE IS HARD

Lessons from OSS in space

- So many constraints
  - Hardware issues: Extreme Temperature variations, Radiation, Pressure (Vacuum), Vibration
  - Limits on: Power, Physical size, Weight (every gram counts)
  - Requirements: Performance, Fault tolerance, Realtime, Power management
- Extremely high cost per mission
  - Low units: often 1 unit
  - High cost of design, testing, hardware, launch, operations
  - Failure is "not an option" (but a high percentage of cubesats fail)
    - This is why craft often last longer than intended
      - Over-engineered, for robustness





Space missions use LOTS of custom hardware

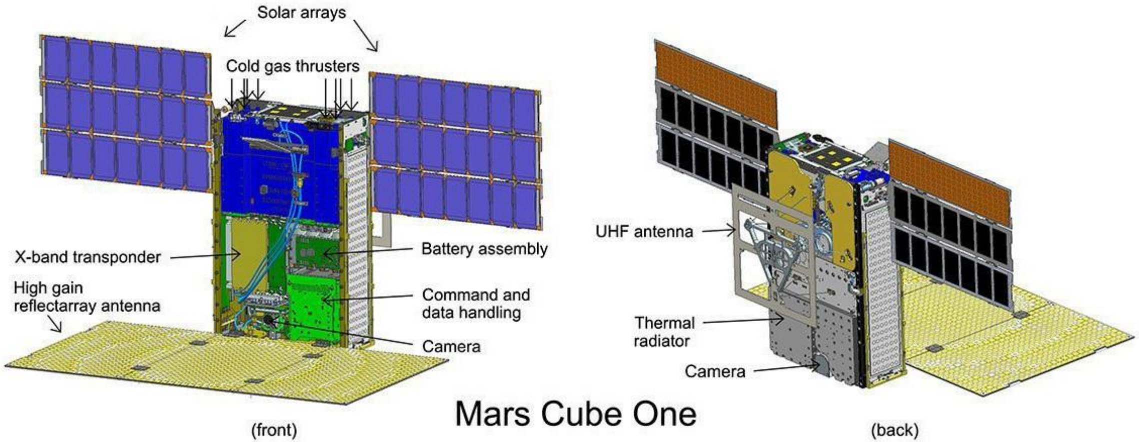
- Focus of mission is specialized science or commercial tasks ('the payload')

  Almost every payload has unique, bespoke hardware

- Even base systems use novel hardware

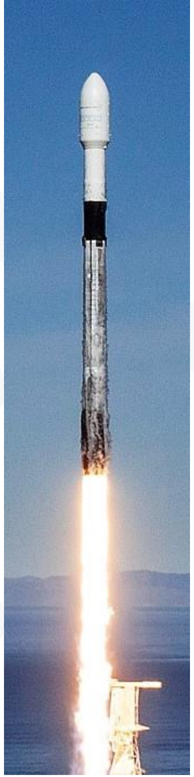
  Thrusters, batteries, stabilizers, power units, sensors, reaction control, etc.

  Every mission seems to want to try something new



Exceptions: COTS hardware and reuse

- SpaceX rockets
  - triple-redundant pairs of COTS x86 processors
- Starlink and Planet satellite constellations
  - x86 processors, not rad-hardened
- Mars Ingenuity helicopter and Perseverance rover and backshell
  - Used some off-the-shelf parts:
    - Qualcomm processor, COTS sensors, USB busses and hubs



Space is embedded in the extreme

- Space sector is "embedded on steroids"
- Emblematic of issues that show up in embedded systems
  - Constraints (power, performance, real-time)
  - Custom-purpose devices and software
  - Hard to find people to collaborate with (for some parts of the stack)

Open Source means collaboration

What defines Open Source?

- Open Source is defined by the ability to use, but also *contribute* to an open code base

- Two effects are key to Open Source

Many Minds Effect

Problem Solver Effect

ELISA Enabling Linux in Safety Applications

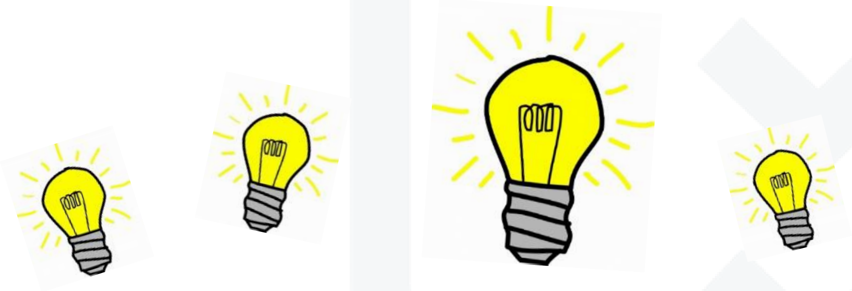
WORKSHOP

The "Many Minds" Effect

Many Minds Effect

- Variety of experiences and skills results in better ideas

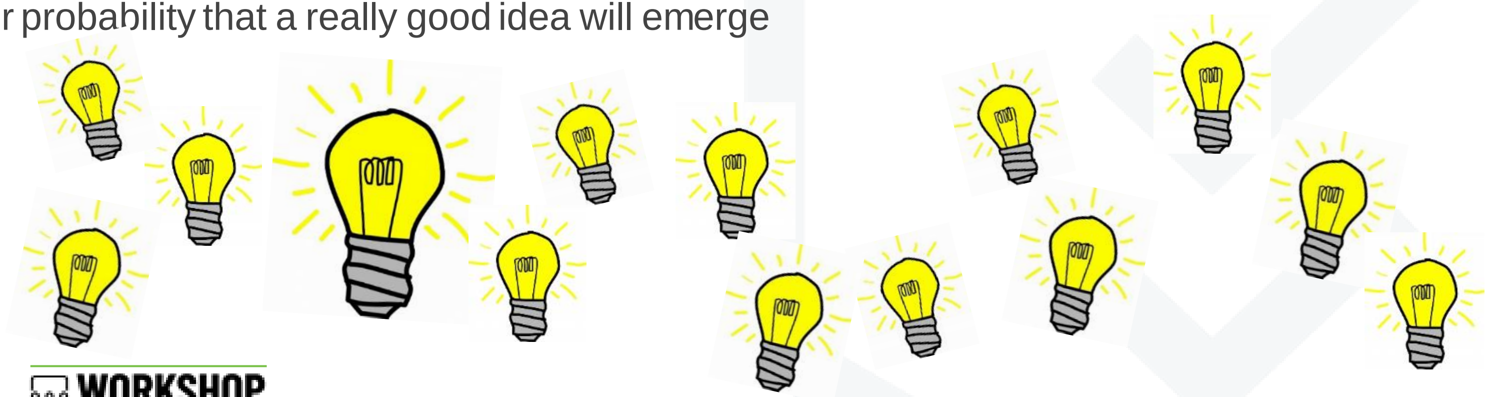
- Open Source strives for a meritocracy, where the best ideas win
- Light bulb analogy:
  - Ideas for a project are like light bulbs...



Open Source effect

- Small community = small number of ideas

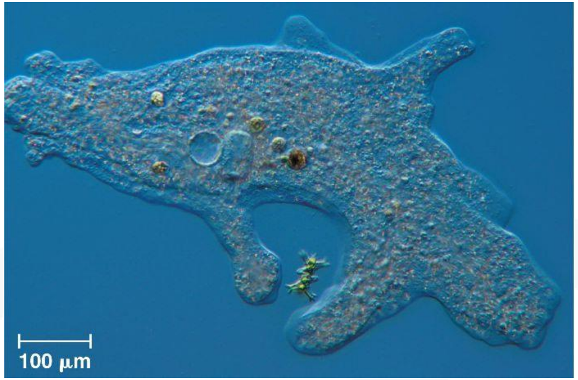
- Bigger community = more ideas
  - Better probability that a really good idea will emerge



Many Minds Effect for bugs

- "Given enough eyeballs, all bugs are shallow"

Problem Solver effect



- Problems are solved as they are encountered

- Software must come “in contact” with a problem space to advance
- Most software is written to solve a specific problem
  - It does not grow outside of it’s original niche
- Openness of OSS allows it to encounter other problem spaces
  - It can adapt and grow in ways different from the original use case

- The OSS virtuous circle: The more problems a piece of software solves, the more users it attracts, and the bigger its community gets

The Paradox of Embedded Open Source

ELISA Enabling Linux in Safety Applications

WORKSHOP

Embedded OSS Paradox

- How to build an ecosystem, when your projects are unique?

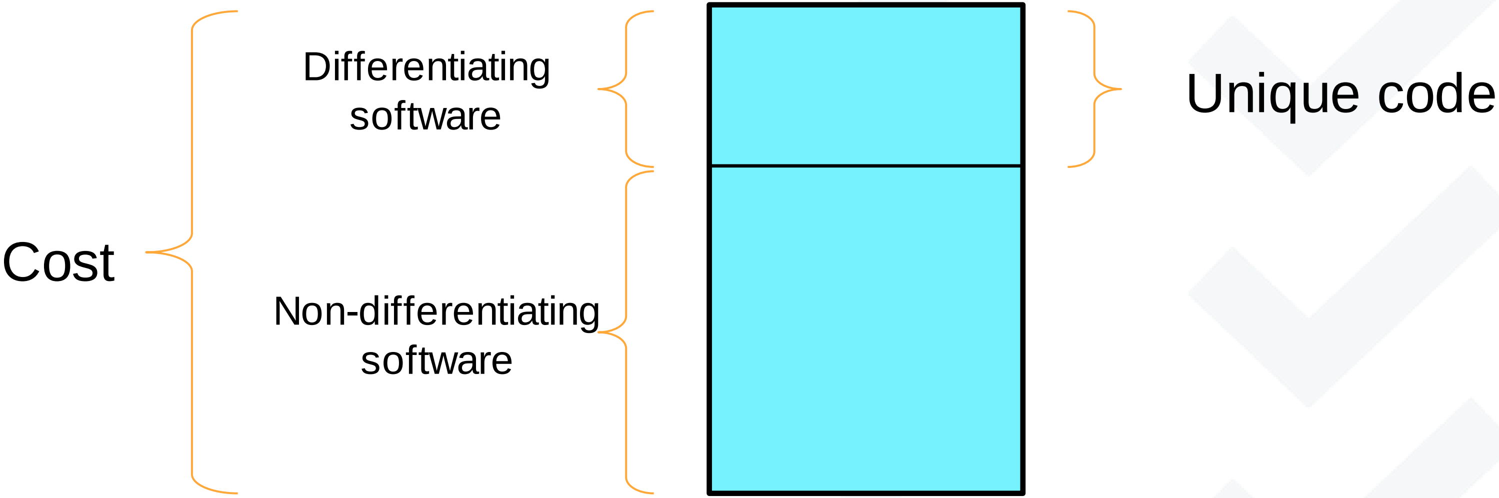
- Other users don't have your use case
- Other users don't use your software
- Other users don't see your bugs
- Your software is not applied to other problem domains

- No Open Source effects!!

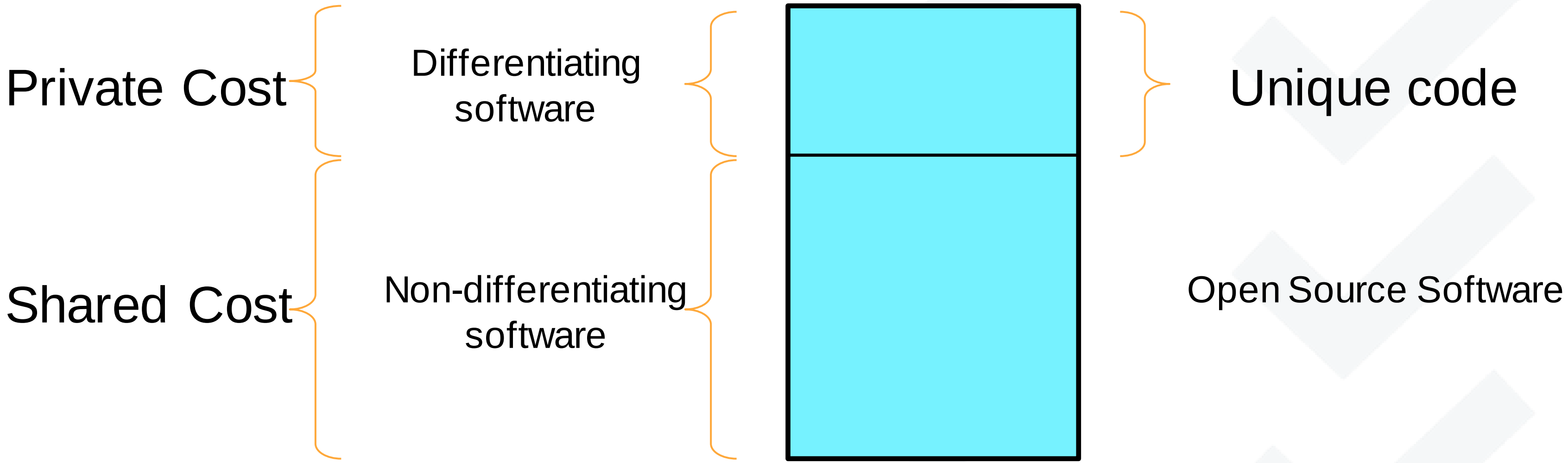
Divide the stack!

- Separate stack into custom solutions and shared code

Differentiating vs. non-differentiating software



Software stack – cheaper way to develop



Private Cost

Differentiating software

Unique code

Shared Cost

Non-differentiating software

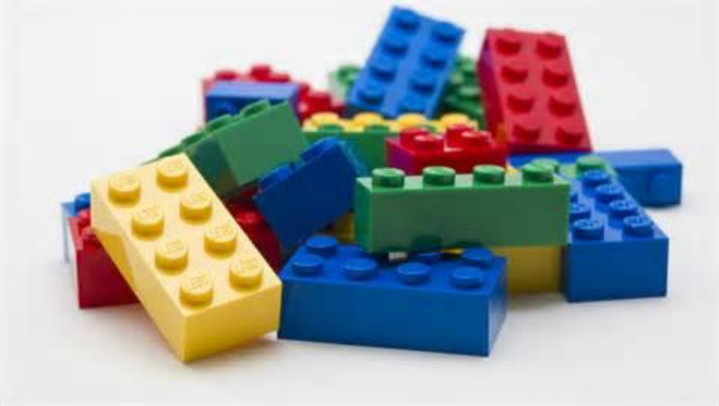
Open Source Software

Generalization vs. Specialization

ELISA Enabling Linux in Safety Applications

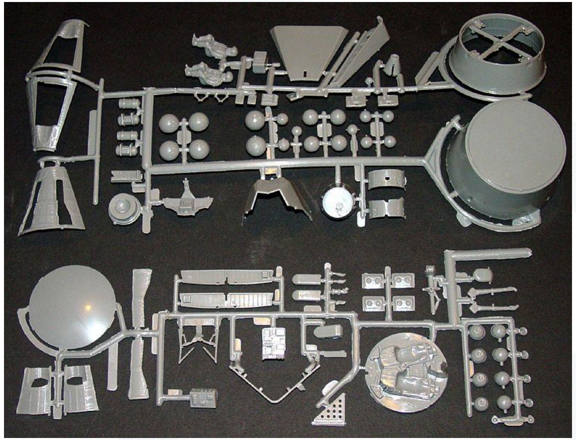
WORKSHOP

Generalization vs. specialization



Legos

- Modular
- Interchangeable
- Reusable



Parts for a model space capsule

- Custom
- Specific
- Fit-for-purpose

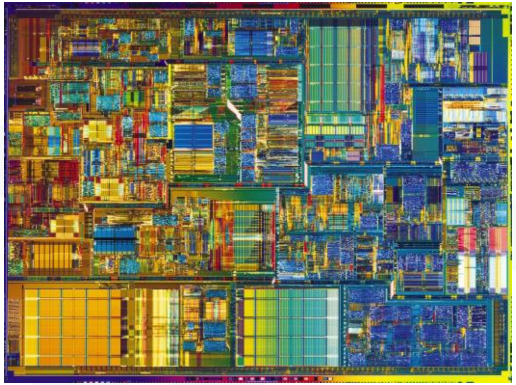
Generalization vs. specialization (cont).

- Spaceship pieces are really good for making a spaceship
- With Lego pieces, you can make also make a spaceship
  - But you can also make a boat, or a car, or a house
- Admittedly, a spaceship made of spaceship pieces will be better
- But the Lego pieces are more general and versatile

Open Source prefer "legos"

ELISA Enabling Linux in Safety Applications WORKSHOP

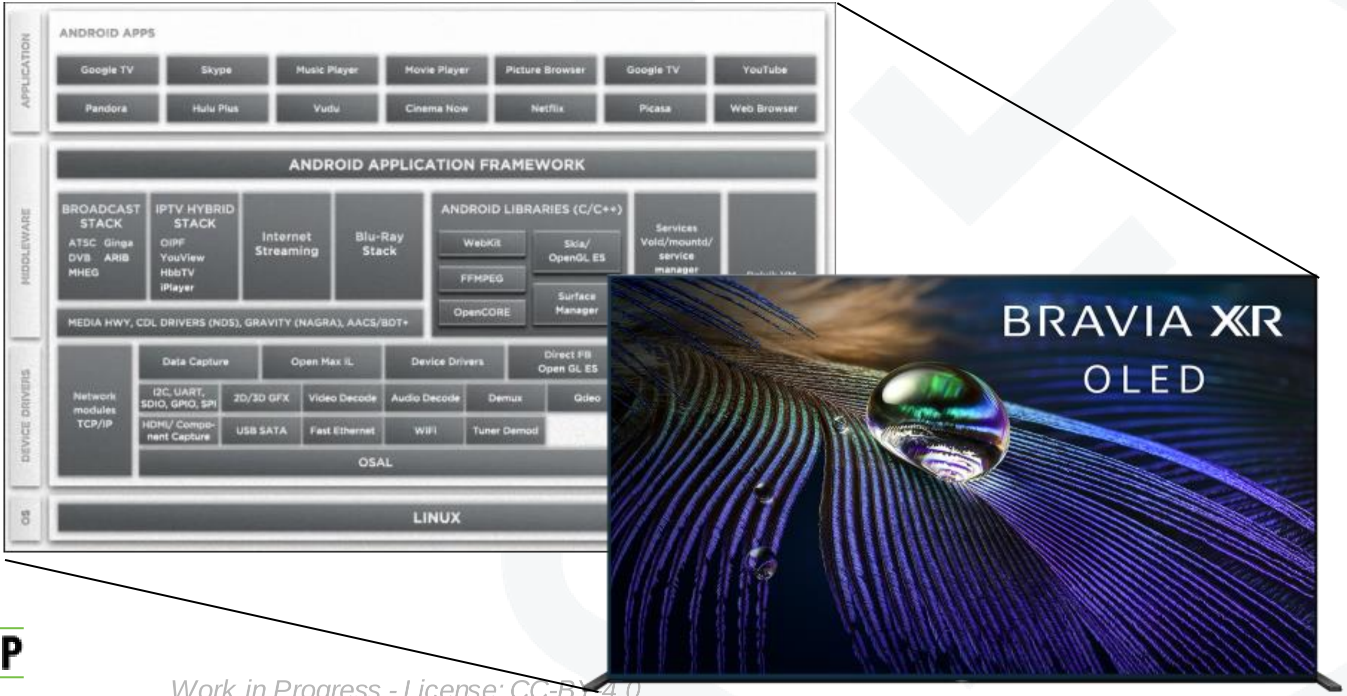
The same is true of modern hardware



- A modern processor has "too much" stuff on it
- Why? – because the processors have been generalized so they can support a wide variety of tasks
  - Commoditization of mobile phone hardware has made processors and hardware features for embedded very cheap
  - There is now a processor that can run Linux, that costs 15 cents
- Your embedded app is unlikely to use every IP block on a modern processor
  - Those are like the rough edges and extra "nubs" on a lego model

Modern software stacks are also complex

- Bravia TV has about 56 million lines of code
- 80 to 90% is open source



Overbuilding, tight margins, and functional safety

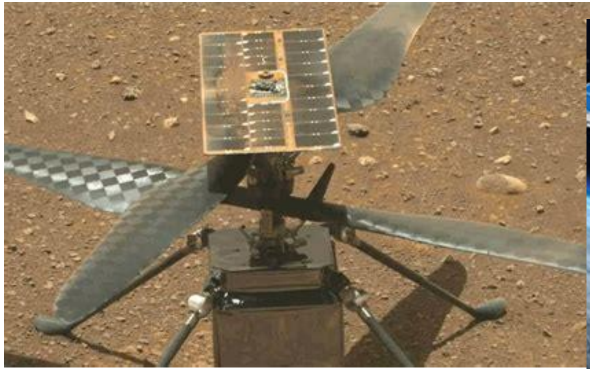
- Does Bravia TV need all that code? (NO)
- Does any embedded product need everything on the SoC? (NO)
- We accept waste (overbuilding) in the processor space but not in the software space

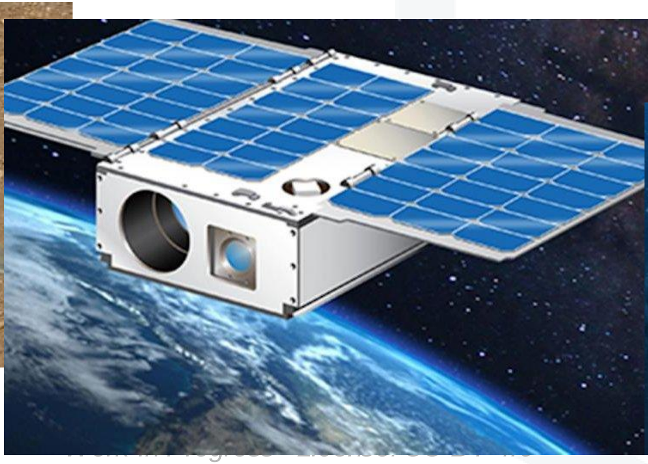
- Functional Safety often means trying to minimize the software to reduce complexity and increase testability

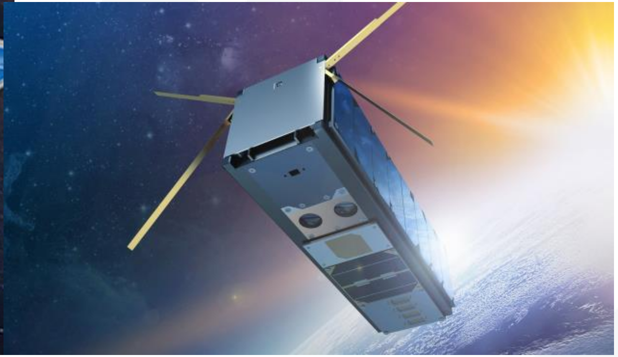
Examples of overbuilding

- Some space missions used shell scripts and Linux distro features to extend capabilities or resolve issues
- Ingenuity used compression to solve a problem, when not in the original plan

  Possible because gzip was in the distribution anyway
- Asteria and Aalto cubesats provided shell callouts







Tips to build an ecosystem

Tips to build an ecosystem

- Increase the community
- Create opportunities for non-experts
- Improve generalization
- Avoid unnecessary specialization
- Find allies

Increase the community

- Actively invite others
- Do something to make the community more interesting or valuable
    - Space has built-in interest factor
    - Gamification
- Reduce barriers to participation
    - Lots of documentation
    - Automation (e.g. project setup tools)
- Contributors come from users

  You have to have users in order to increase the pool of contributors

Create opportunities for contribution by non-experts

- Contributions can be in many forms

  Usage reports

  Bug reports

  Documentation

  Infrastructure management

  Testing

  Reviewing

  Marketing and advocacy

  Code

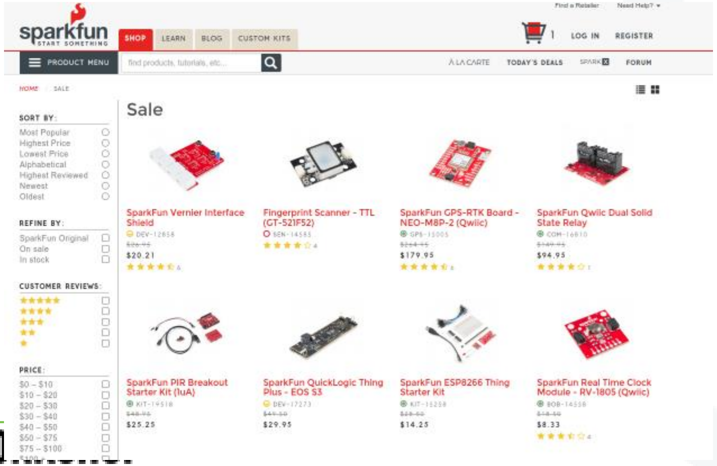
Improve generalization

- Extend existing mechanisms rather than add new ones
  - Candidate: Use a Linux IPC instead of your own message bus
- Make sure your contributions handle other people's use cases

Avoid unnecessary specialization

- Use the same hardware that others are using
- Use the same sub-systems and software technologies as others
- Don't over-reduce

Ship with more than the absolute minimum you need



Sale

SORT BY:
- Most Popular ○
- Highest Price ○
- Lowest Price ○
- Alphabetical ○
- Highest Reviewed ○
- Newest ○
- Oldest ○

REFINE BY:
- SparkFun Original □
- On sale □
- In stock □

CUSTOMER REVIEWS:
- ★★★★★ □
- ★★★★ □
- ★★★ □
- ★★ □
- ★ □

PRICE:
- $0 – $10 □
- $10 – $20 □
- $20 – $30 □
- $30 – $40 □
- $40 – $50 □
- $50 – $75 □
- $75 – $100 □
- $100 + □

SparkFun Vernier Interface Shield
DEV-12858
~~$26.95~~
$20.21
★★★★☆ 6

Fingerprint Scanner - TTL (GT-521F52)
SEN-14585
★★★★☆ 4

SparkFun GPS-RTK Board - NEO-M8P-2 (Qwiic)
GPS-15005
~~$264.95~~
$179.95
★★★★☆ 6

SparkFun Qwiic Dual Solid State Relay
COM-16810
~~$149.95~~
$94.95
★★★★☆ 1

SparkFun PIR Breakout Starter Kit (1uA)
KIT-19518
~~$48.95~~
$25.25

SparkFun QuickLogic Thing Plus - EOS S3
DEV-17273
~~$49.50~~
$29.95

SparkFun ESP8266 Thing Starter Kit
KIT-15258
~~$28.50~~
$14.25

SparkFun Real Time Clock Module - RV-1805 (Qwiic)
BOB-14558
~~$18.50~~
$8.33
★★★☆☆ 4

Find technical allies

- Find people who care about your issues, and work with them
- Sometimes, it's not who you expect:
  - Small system size
    - Security researchers interested in reduced attack surface
    - Cloud service companies (for low-footprint VMs)
  - Low power usage
    - Mobile phone developers, IOT developers
    - Data Center Linux developers
  - Fault Tolerance
    - Banking, Routers

Technical Allies – boot time example

- Recently started a Boot Time Special Interest Group (SIG)
- Found lots of people from different sectors interested:
  - Automotive, Consumer Electronics, Desktop, Mobile
  - Some unexpected: Cloud Servers, Supercomputers
    - For quick service spinup, initialization of chips with high CPU count
  - Lots of developers with limited kernel development experience
    - I created automated tools and a wiki for people to contribute data and docs

Conclusion

Let's work together on

a bright and interesting future!

By NASA - https://www.nasa.gov/sites/default/files/atoms/files/artemis_plan-20200921.pdf, Public Domain

Thanks!

Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [https://creativecommons.org/licenses/by/4.0/] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.