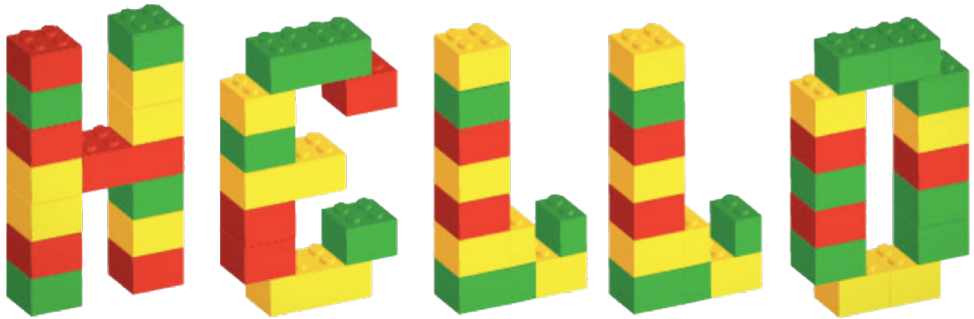
The Ara System Architecture

Alex Elder
Linux Technical Lead: Project Ara
Linaro

Introduction

- Ara is a modular phone being developed by the ATAP group at Google
- Modularity offers advantages for developers and consumers
- Supporting modularity on an embedded device has challenges
- This talk will show the Ara project is addressing some of them



Project Ara: The Buzz

Much has been said about Project Ara, a lot (but not all) of it positive and hopeful.

"An innovative, modular smartphone"

"Everything is awesome!"

"Strange and beautiful"

"Destined to fail"

"The ultimate Swiss Army knife"

"A pipe dream"

"The true phone of the future"

"This will never work"

"El smartphone que revolucionará el mercado"

Project Ara: The Reality

Project Ara is:

- More than a “modular phone”
- A modular **platform** (and not just one phone) being designed
- A developing ecosystem and marketplace for modules

Still...

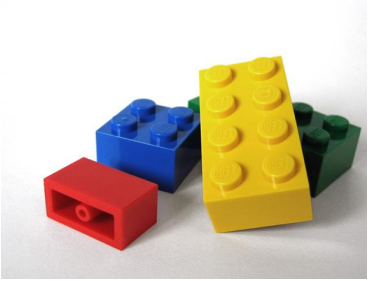
This talk will stay focused on technical issues

Why a modular phone?

- Buyers can configure and reconfigure their devices
- Integrated phone can't justify including non-mainstream features
- Components can be developed independent of the “base” phone
- Modularity is designed in (unlike external cables/dongles)
- Smaller unit of repair/replacement

Creating a modular device is hard...

- Conventional designs assume things never change
- Specific features supported aren't known at design time
- Can't predict what future features a module might provide
- Specific power or data requirements of modules not known
- System must handle dynamic addition/removal of modules



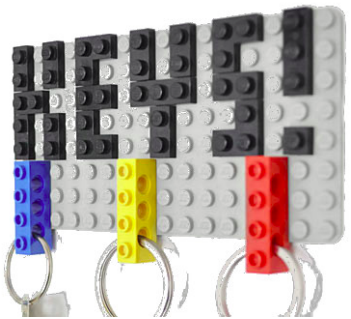
Requirements for modularity

To allow for modularity, well-defined interfaces are needed:

- Physical (size, shape, durability)
- Electrical (power, communication)
- Software (configuration, operation, and user interaction)

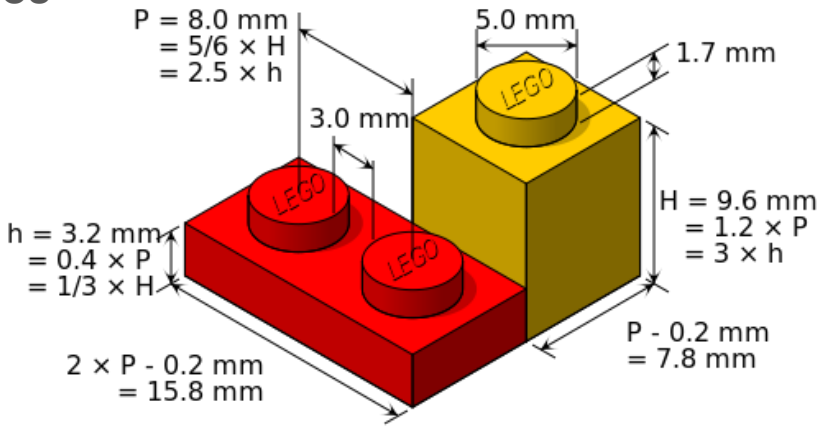
These interfaces must allow things to change at run time

Modules should also not look like they were an afterthought



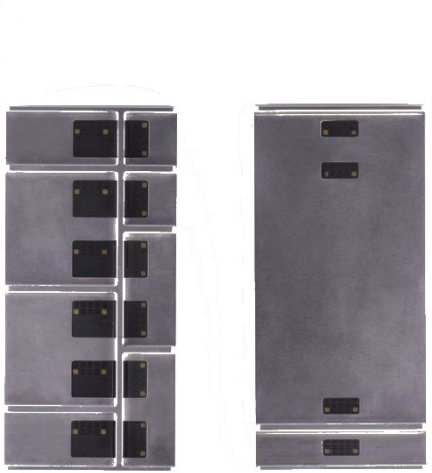
Physical interface requirements

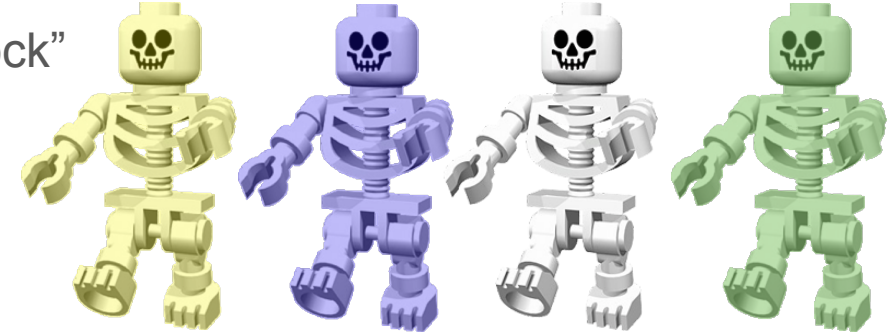
- Support removable modules (and tolerate repeated removal)
- Hold module securely (#DropTest, #Thieves)
- Support a reasonable module size--large (or small) enough
- Tolerate the special abuses a phone endures



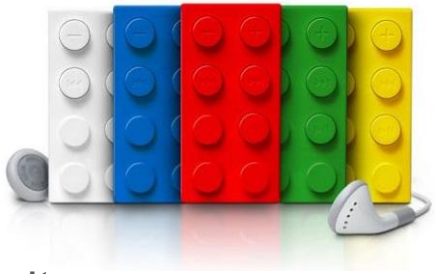
Ara Endoskeleton

- Rigid substrate, but as lightweight as possible
- Physical guides hold modules in place
- Electrically controlled mechanism prevents removal
- Slots available in several module sizes
- Each slot has an electrical “interface block”





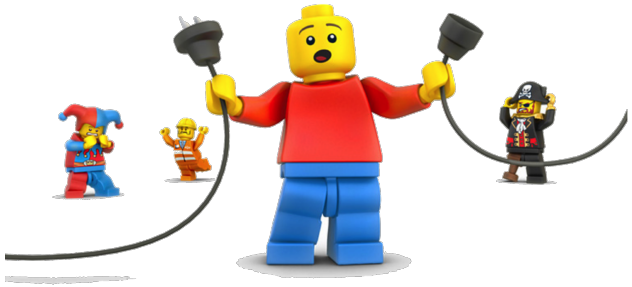
Ara modules



- Contain units of functionality (module devices)
- Multiple standard sizes support modules of varying complexity
- Changeable skins allow user customization

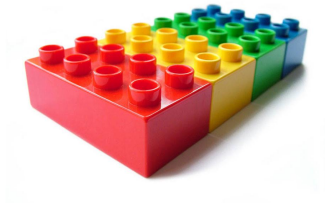
Electrical interface requirements

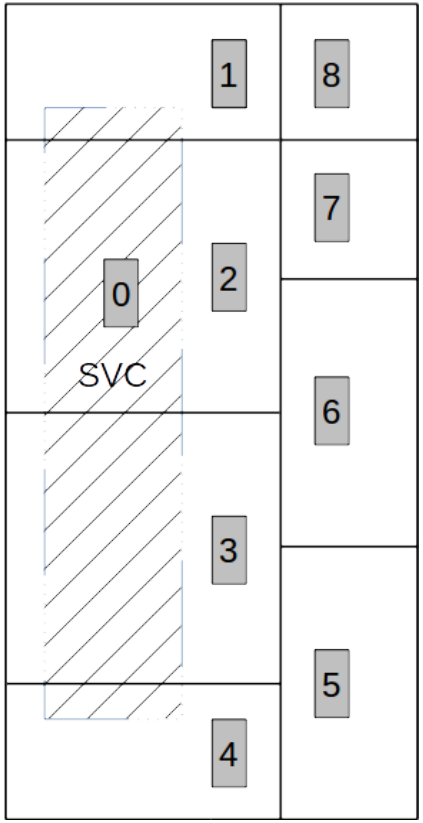
- Safely provide power for modules
- Prevent excessive module power consumption
- Provide a usable data path while managing power consumption
- Enable the system to know when a module is present



Ara electrical design

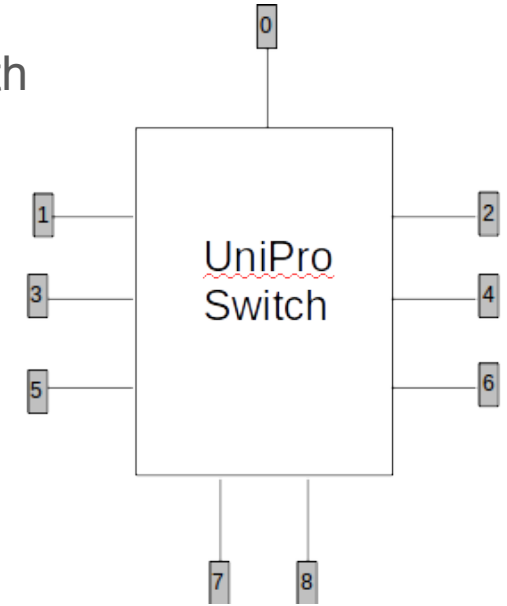
- Endoskeleton contains a "Service Control" processor (SVC)
- Module positions contain numbered “interface blocks”
- Interface block provides presence detect logic to the SVC
- Interface block also provides power to module
- SVC controls power to each interface block (and limits it)





Ara electrical design (continued)

- Endoskeleton also contains a MIPI® UniPro℠ switch
- Each interface block has a UniPro switch connection
- UniPro network provides high speed, low power data path
- SVC sits on UniPro switch as well (position 0)



UniPro

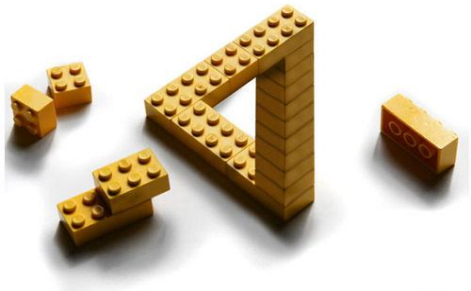
- Bidirectional multi-lane links transfer data at up to 5 Gbps per lane
- Reliable, in-order transfer of data over the link
- Each link supports multiple independent connections
- Links can run at lower data rates to reduce power consumption

Software interface requirements

- Allow for the discovery of module features and requirements
- Allow system software to communicate with control module devices
- Support new/future device types
- Provide a way for user to release modules
- Provide a way to update module firmware

Ara software design

- SVC firmware works with kernel to manage the Endoskeleton
- Kernel identifies modules, connects module devices with their drivers
- Drivers work with module firmware to implement device functionality
- Android works with the kernel to handle dynamic devices
- Ara manager Android app provides user control over modules
- Online resources supply module support packages
- If required, firmware is updated via UniPro

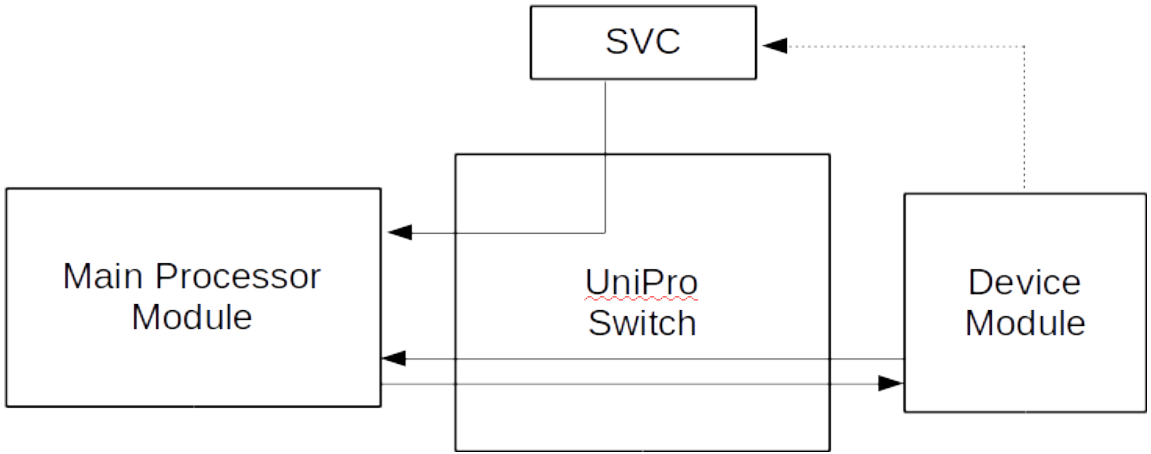


Greybus

- Defines messages sent over connections between modules
- Supports “operations” that pair a request and response message
- Protocols define sets of operations a connection supports
- Protocols implement classes of device behavior
- Modules advertise classes they support in a "manifest"

Putting it all together

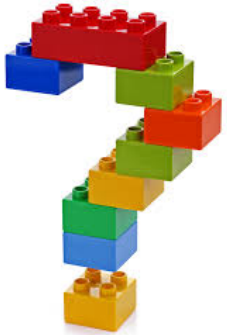
- Module insertion causes SVC to notify main processor
- Kernel on main processor gets manifest from module
- Kernel notifies Android, allowing support code to be downloaded
- Android app initiates module removal



Summary

Ara is a modular device platform that:

- Acts as a phone, but supports add-on modules
- Detects module insertion and controls module removal
- Manages power to modules independently
- Provides a UniPro network, allowing power and performance to be balanced
- Uses a manifest to describe module capabilities
- Leverages online sources to support modules if required



Questions?