goto; conference aarhus

THE ACTOR MODEL APPLIED TO THE RASPBERRY PI AND THE EMBEDDED DOMAIN

Omer Kilic || @OmerK

Erlang Solutions Ltd.

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

Agenda

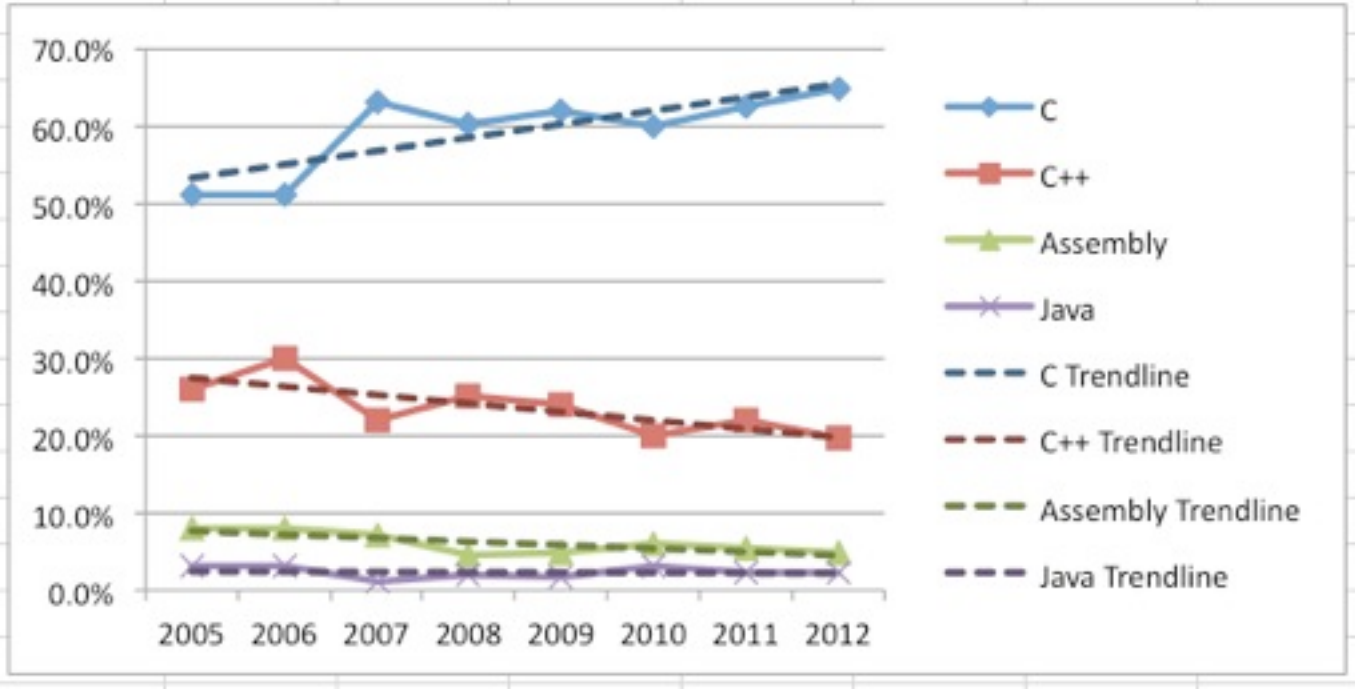
- Current state of Embedded Systems
- Overview of the Actor Model
- Erlang Embedded Project
- Modelling and developing systems using Erlang
- Experiments with the Raspberry Pi
- Future Explorations
- Q & A

Embedded Systems

> An embedded system is a computer system designed for specific control functions within a larger system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs
>
> – Infinite Wisdom of Wikipedia

#include "stats.h"

The four languages most often reported as the primary language for embedded projects for the years 2005 to 2012, along with linear trendlines.



Source: http://embedded.com/electronics-blogs/programming-pointers/4372180/Unexpected-trends

Current Challenges

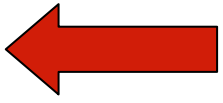
- Complex SoC platforms
- "Internet of Things"
  - Connected and distributed systems
- Multicore and/or heterogeneous devices
- Time to market constraints

Erlang SOLUTIONS

Embedded Systems

- Bare Metal
  - No underlying OS or high level abstractions
- RTOS
  - Minimal interrupt and thread switching latency, scheduling guarantees, minimal jitter
- Embedded Linux
  - Slimmed down Linux, with hardware interfaces

Embedded Systems

- Bare Metal
  - No underlying OS or high level abstractions
- RTOS 
  - Minimal interrupt and thread switching latency, scheduling guarantees, minimal jitter
- Embedded Linux 
  - Slimmed down Linux, with hardware interfaces

RTOS Concepts

- Notion of "tasks"
- OS-supervised interprocess messaging
  - Shared memory
- Mutexes/Semaphores/Locks
- Scheduling
  - Pre-emptive: event driven
  - Round-robin: time multiplexed

Embedded Linux

- Not a new concept, increased popularity due to abundant supply of cheap boards
  - Raspberry Pi, Beagleboard/Beaglebone, Gumstix et al.
- Familiar set of tools for software developers, new territory for embedded engineers
  - No direct mapping for RTOS concepts, expecially tasks
- Complex device driver framework
  - Here be dragons

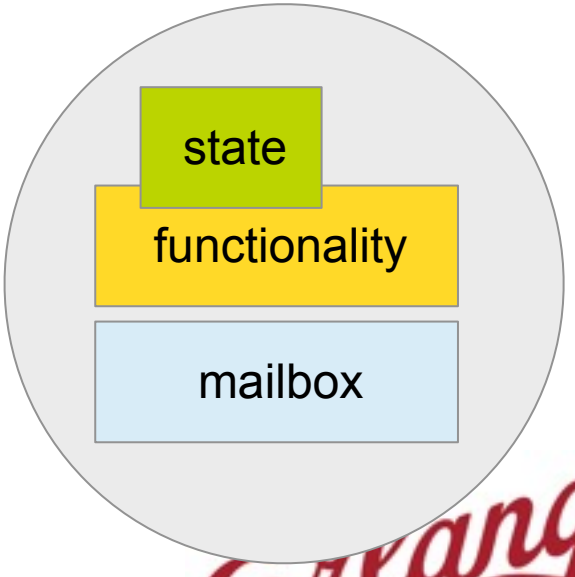
Erlang SOLUTIONS

Actor Model

- Proposed in 1973 by Hewitt, Bishop, and Steiger
  - "Universal primitives of concurrent computation"
- Building blocks for modular, distributed and concurrent systems
- No shared-state, self-contained and atomic
- Implemented in a variety of programming languages

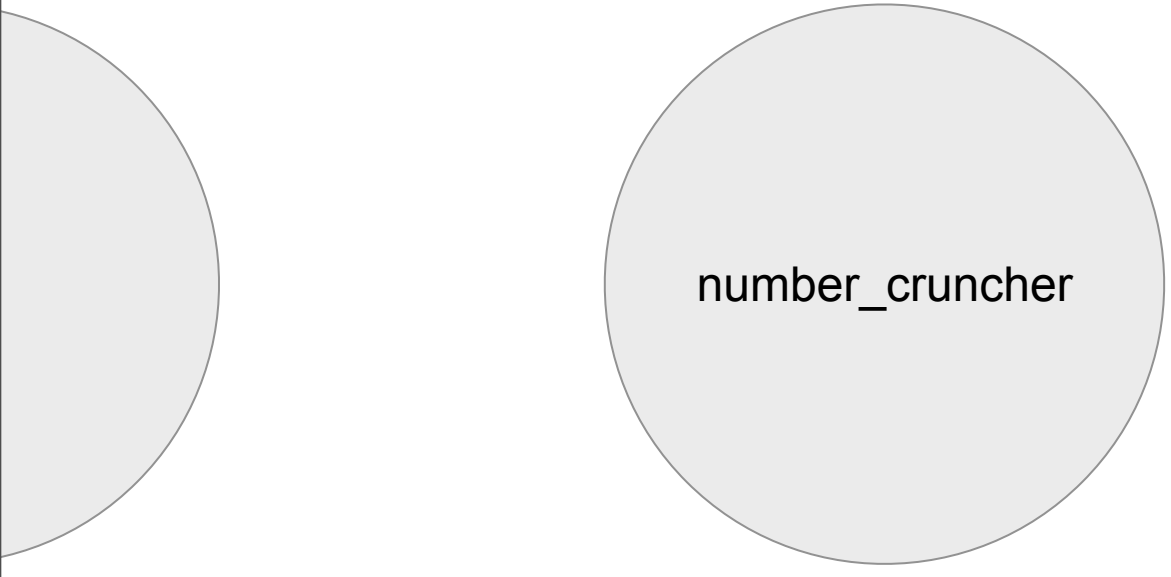
Actor Model

- Asynchronous message passing
  - Messages kept in a mailbox and processes in the order they are received in
- Upon receiving a message, actors can:
  - Make local decisions and change internal state
  - Spawn new actors
  - Send messages to other actors

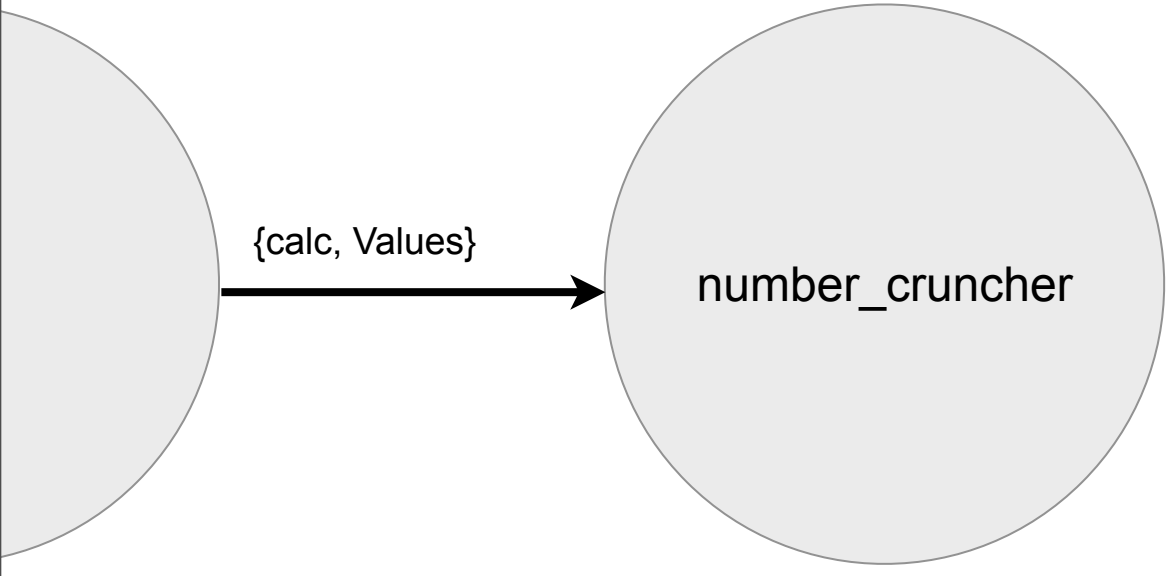


Actor Model

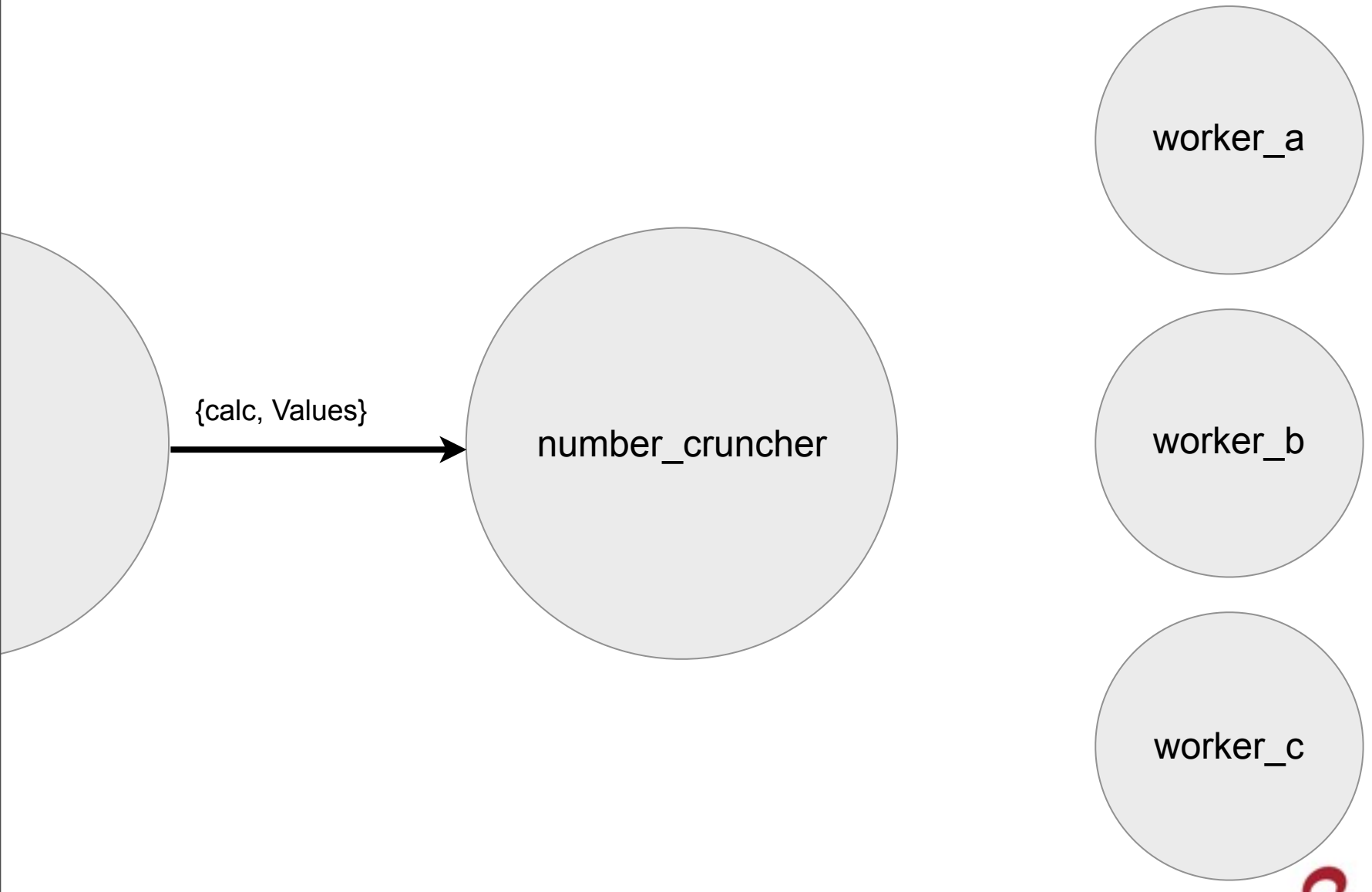
Actor Model



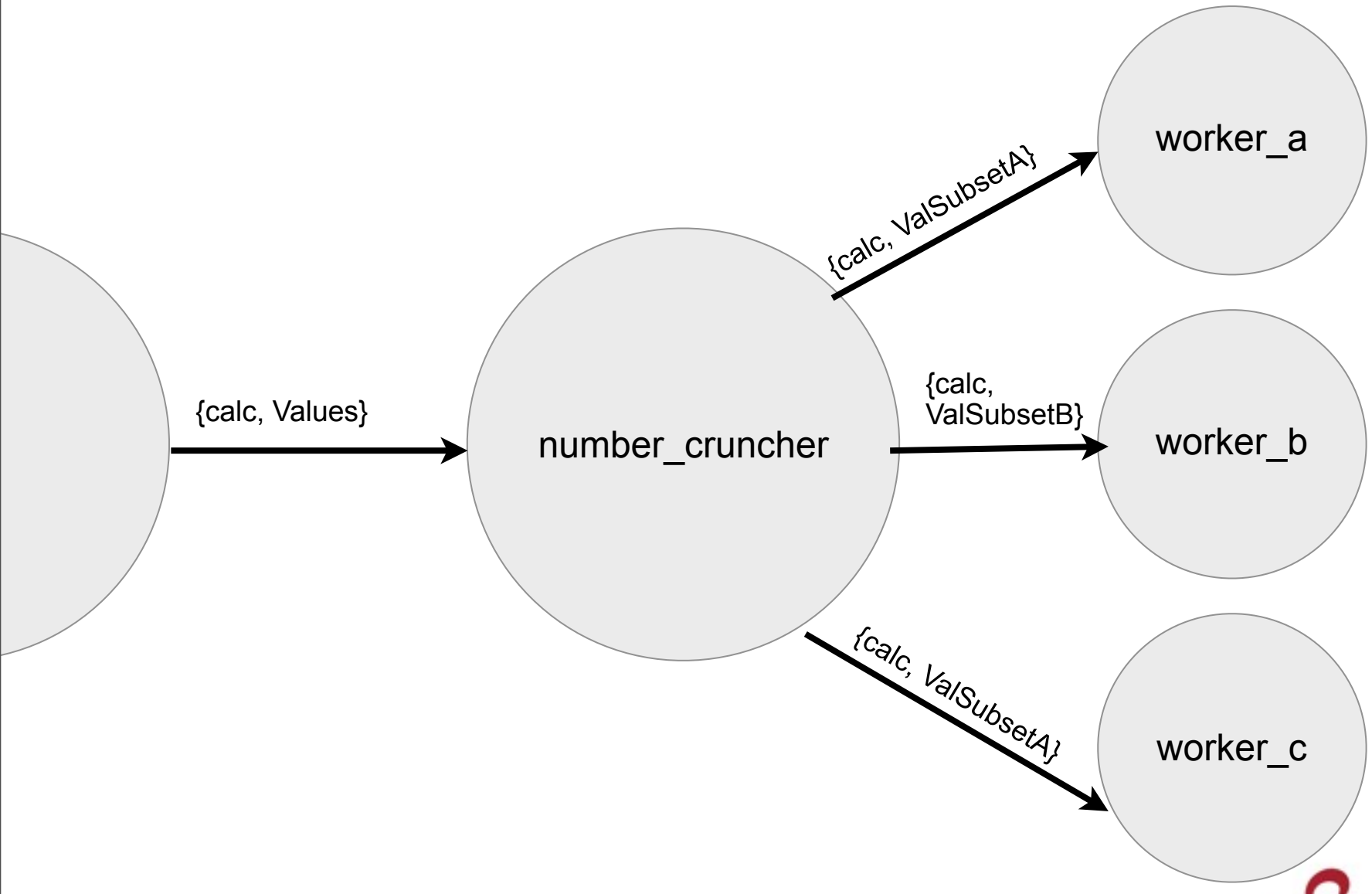
Actor Model



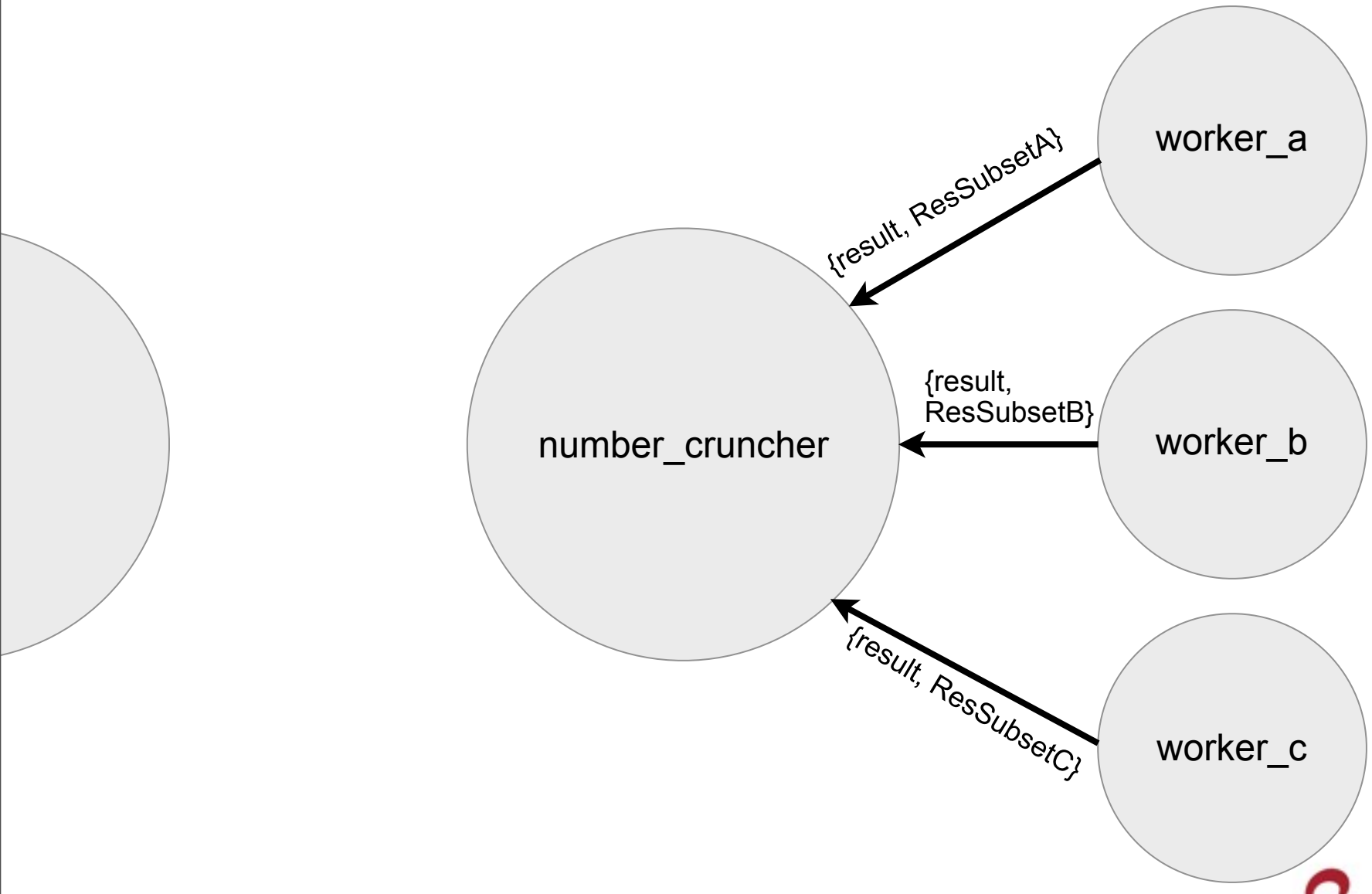
Actor Model



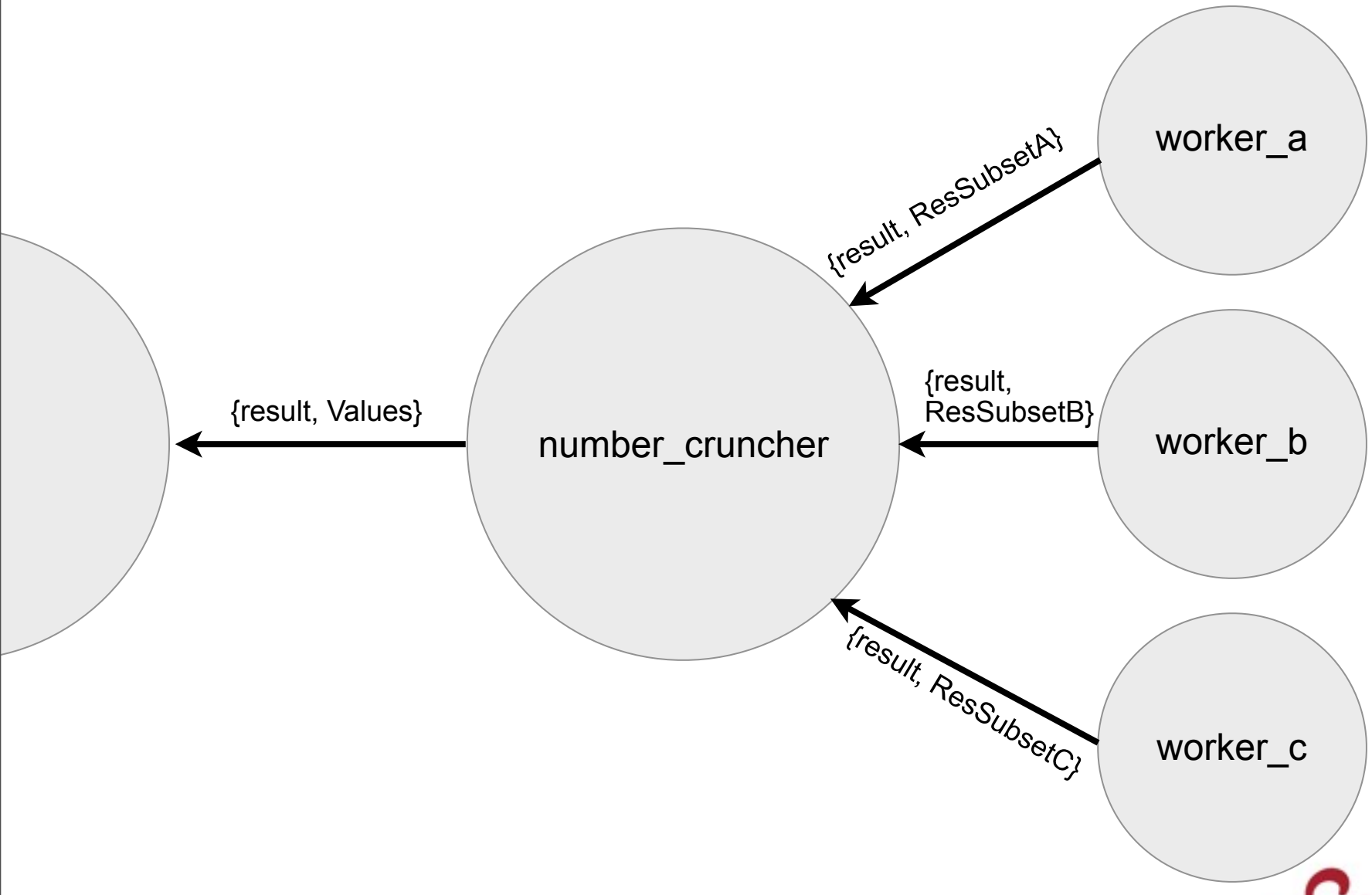
Actor Model


{calc, Values}
number_cruncher
{calc, ValSubsetA}
{calc, ValSubsetB}
{calc, ValSubsetA}
worker_a
worker_b
worker_c

Actor Model


worker_a
{result, ResSubsetA}
number_cruncher
{result,
ResSubsetB}
worker_b
{result, ResSubsetC}
worker_c

Actor Model


worker_a
{result, ResSubsetA}
{result, Values}
number_cruncher
{result, ResSubsetB}
worker_b
{result, ResSubsetC}
worker_c

Limitations of the Actor Model

- No notion of inheritance or general hierarchy
  - Specific to the language and library implementation
- Asynchronous message passing can be problematic for certain applications
  - Ordering of messages received from multiple processes
- Abstract definition may lead to inconsistency in larger systems
  - Fine/Coarse Grain

Erlang Embedded

- Knowledge Transfer Partnership between Erlang Solutions and University of Kent
  - Aim of the project: Bring the benefits of concurrent systems development using Erlang to the field of embedded systems; through investigation, analysis, software development and evaluation.
- http://erlang-embedded.com

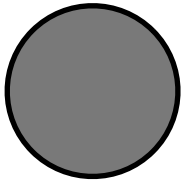
Why Erlang

- Implements the Actor model
- Battle-tested at Ericsson and many other companies
  - Originally designed for embedded applications
- Support for concurrency and distributed systems out of the box
- Easy to create robust systems
- (...and more!)

High Availability / Reliability

- Simple and consistent error recovery and supervision hierarchies
- Built in fault-tolerance
  - Isolation of Actors
- Support for dynamic reconfiguration
  - Hot code loading

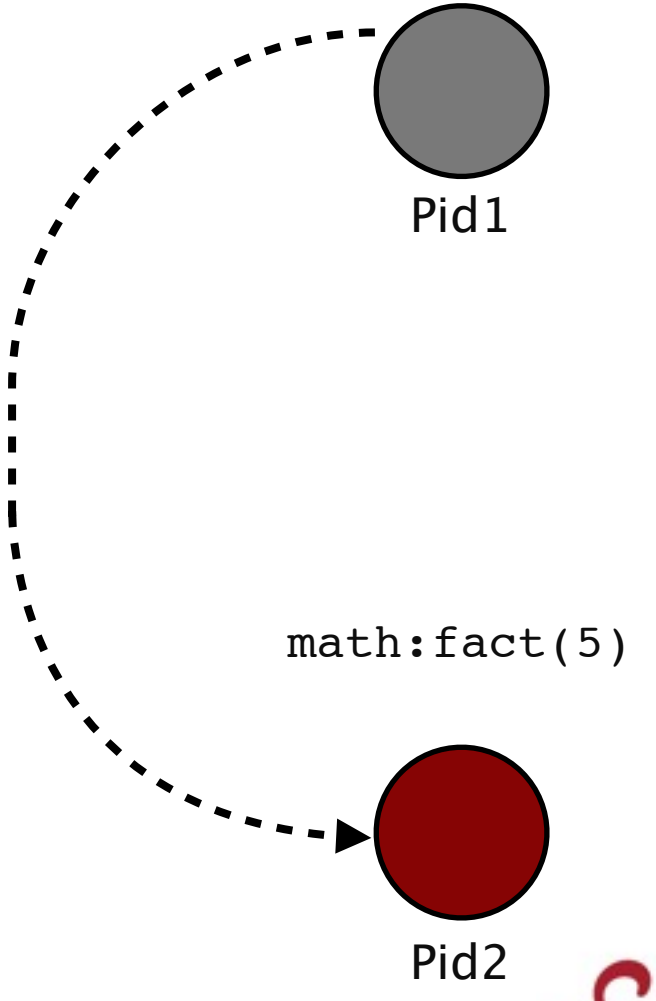
Creating an Actor




```
spawn(math, fact, [5])
```

Creating an Actor

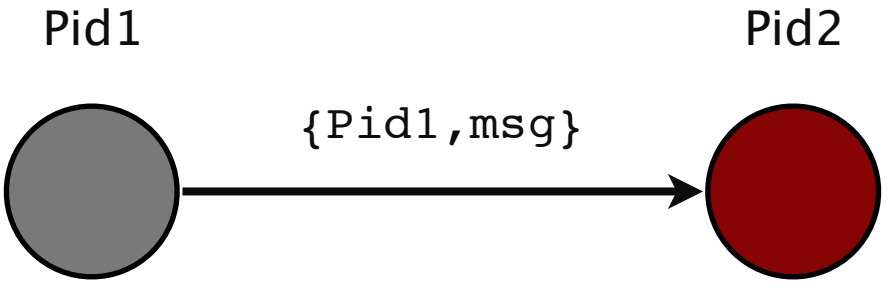
```
spawn(math, fact, [5])
```



Communication


```
Pid2 ! {self(),msg}
```

Communication




```
Pid2 ! {self(),msg}
```

Bidirectional Links

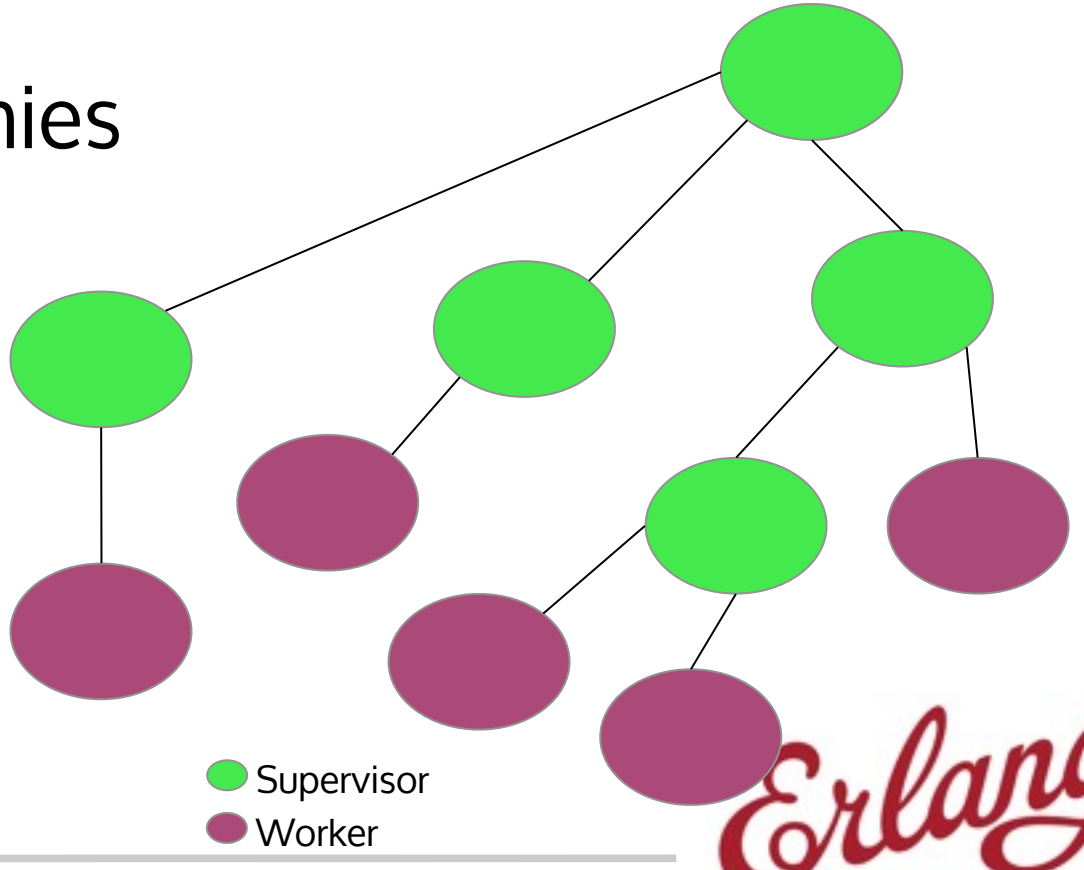

```
link(PidB)
```

Bidirectional Links

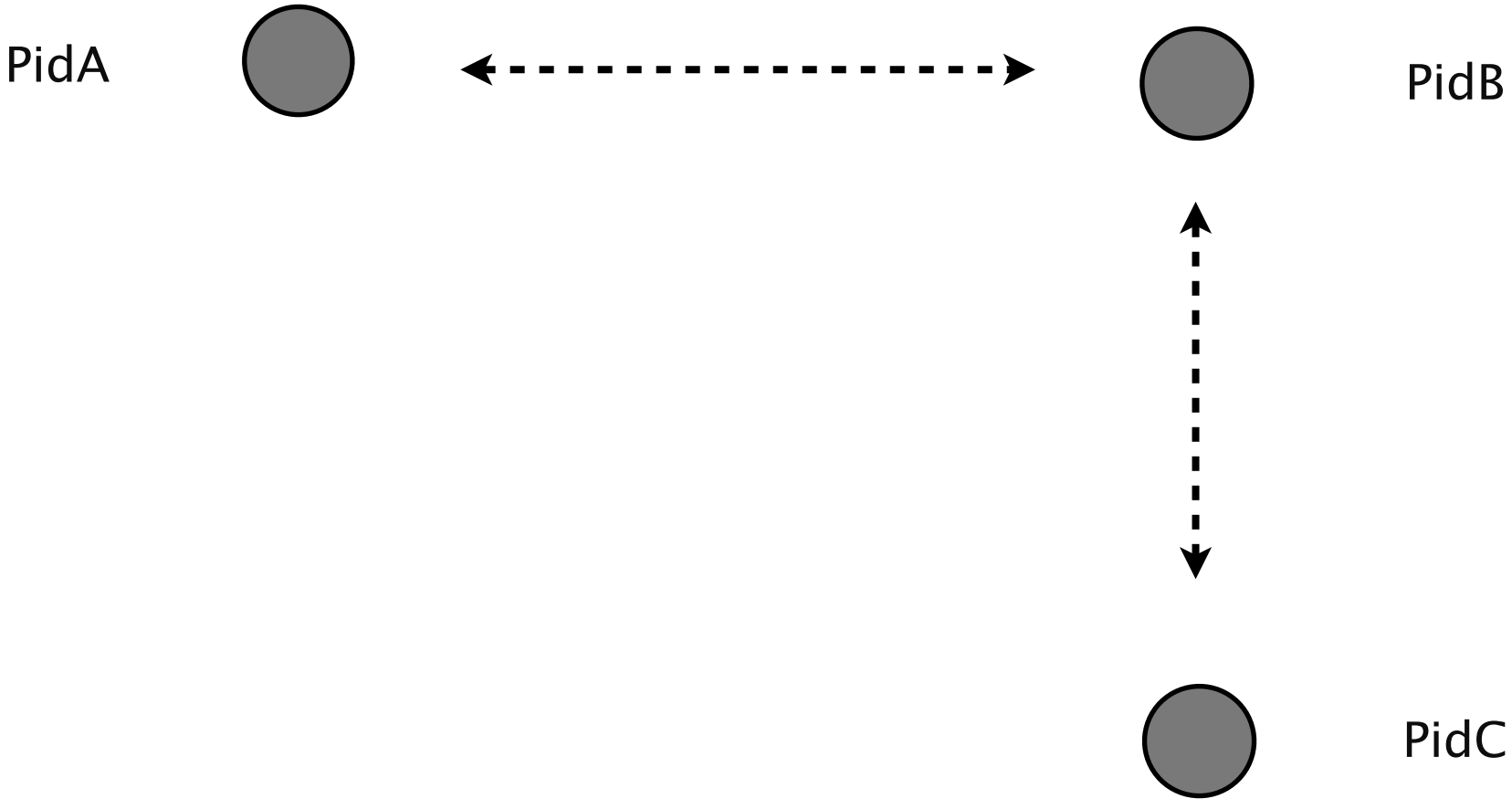

```
link(PidB)
```

Process Error Handling

- Let it Fail!
  - Abstract error handling away from the modules
  - Leaner modules
- Supervision hierarchies

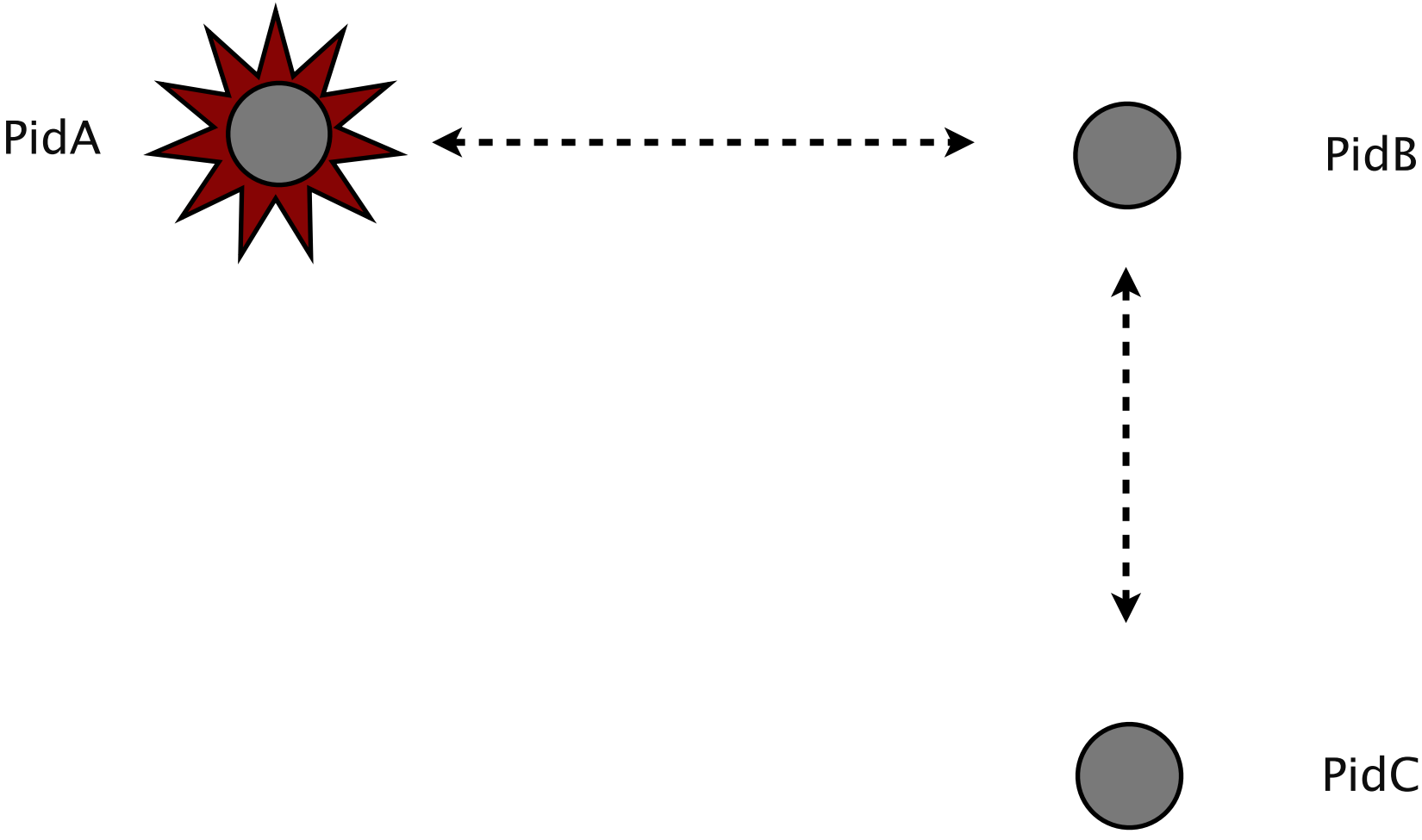


Propagating Exit Signals



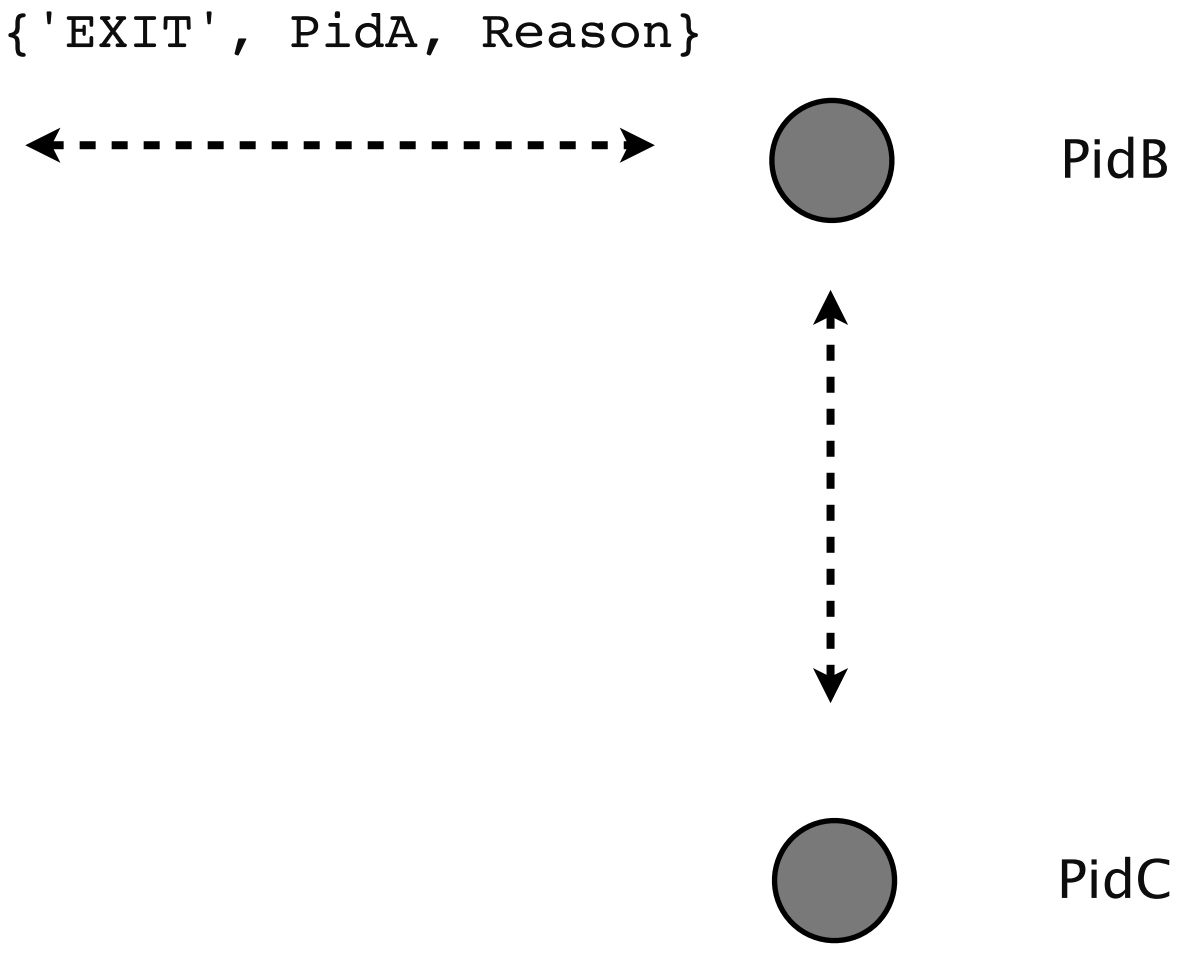
Erlang SOLUTIONS

Propagating Exit Signals

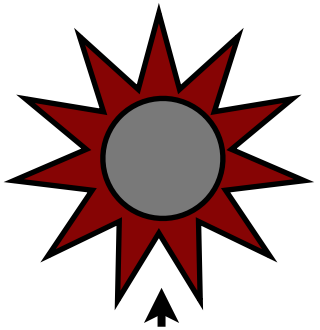


Erlang SOLUTIONS

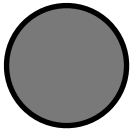
Propagating Exit Signals



Propagating Exit Signals



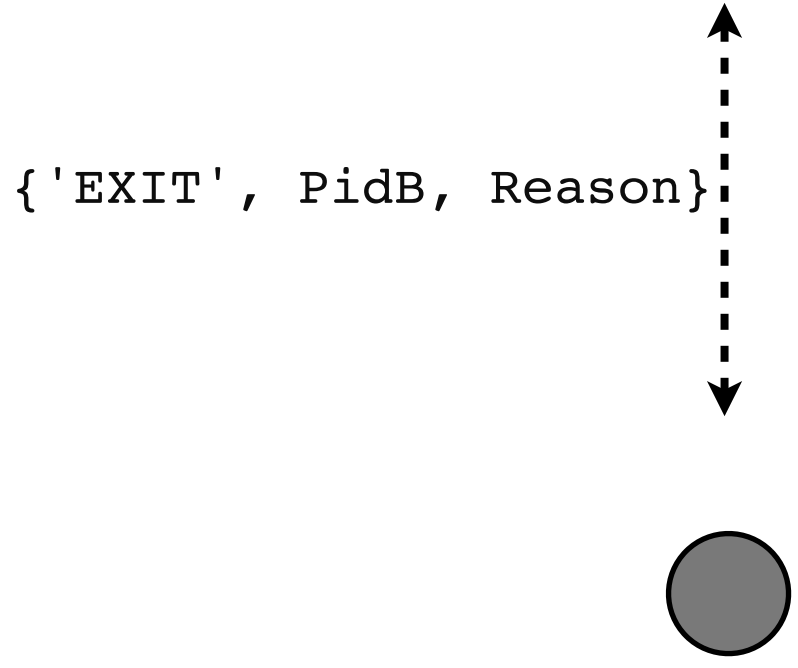
PidB



PidC

Erlang SOLUTIONS

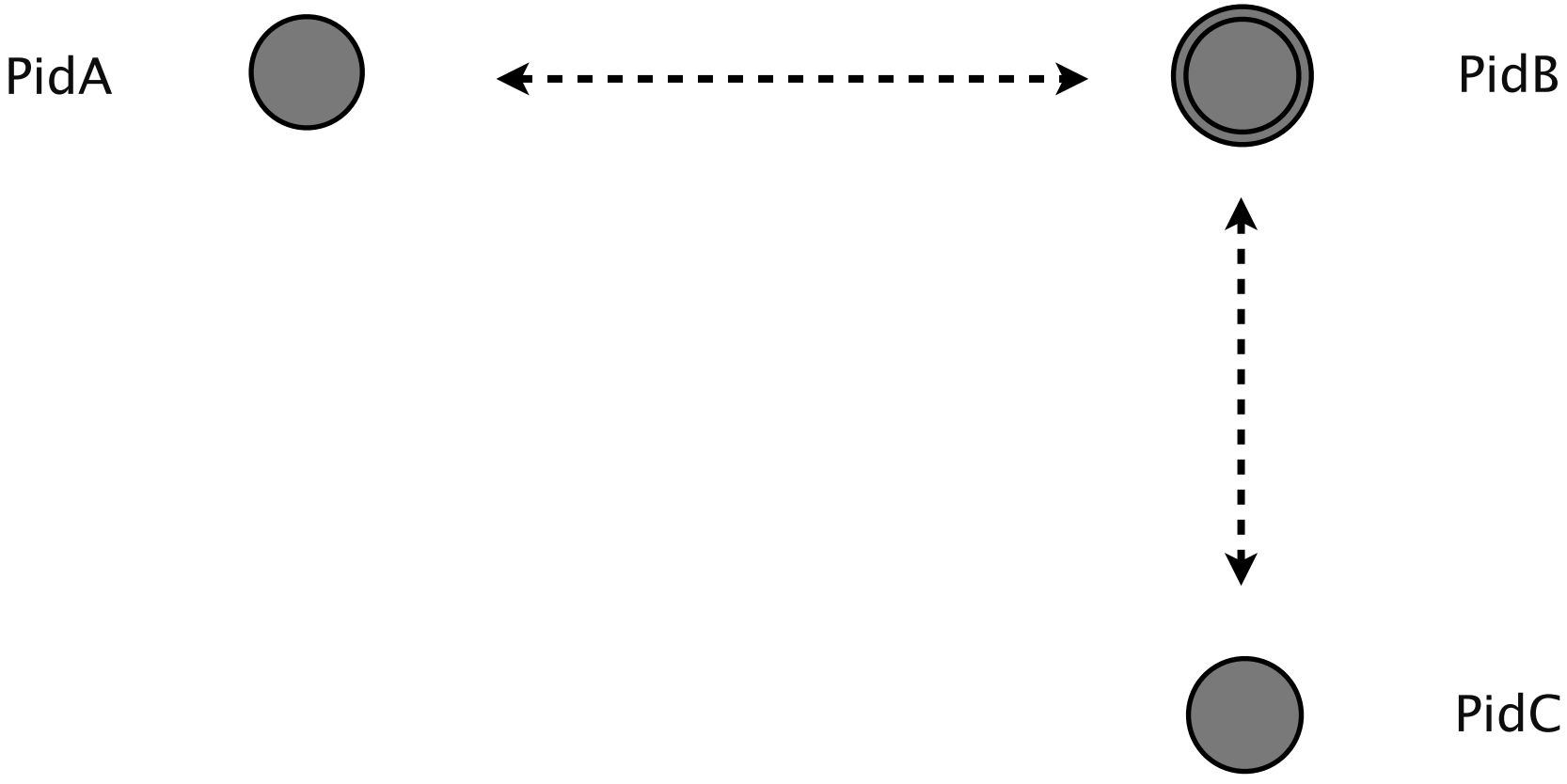
Propagating Exit Signals



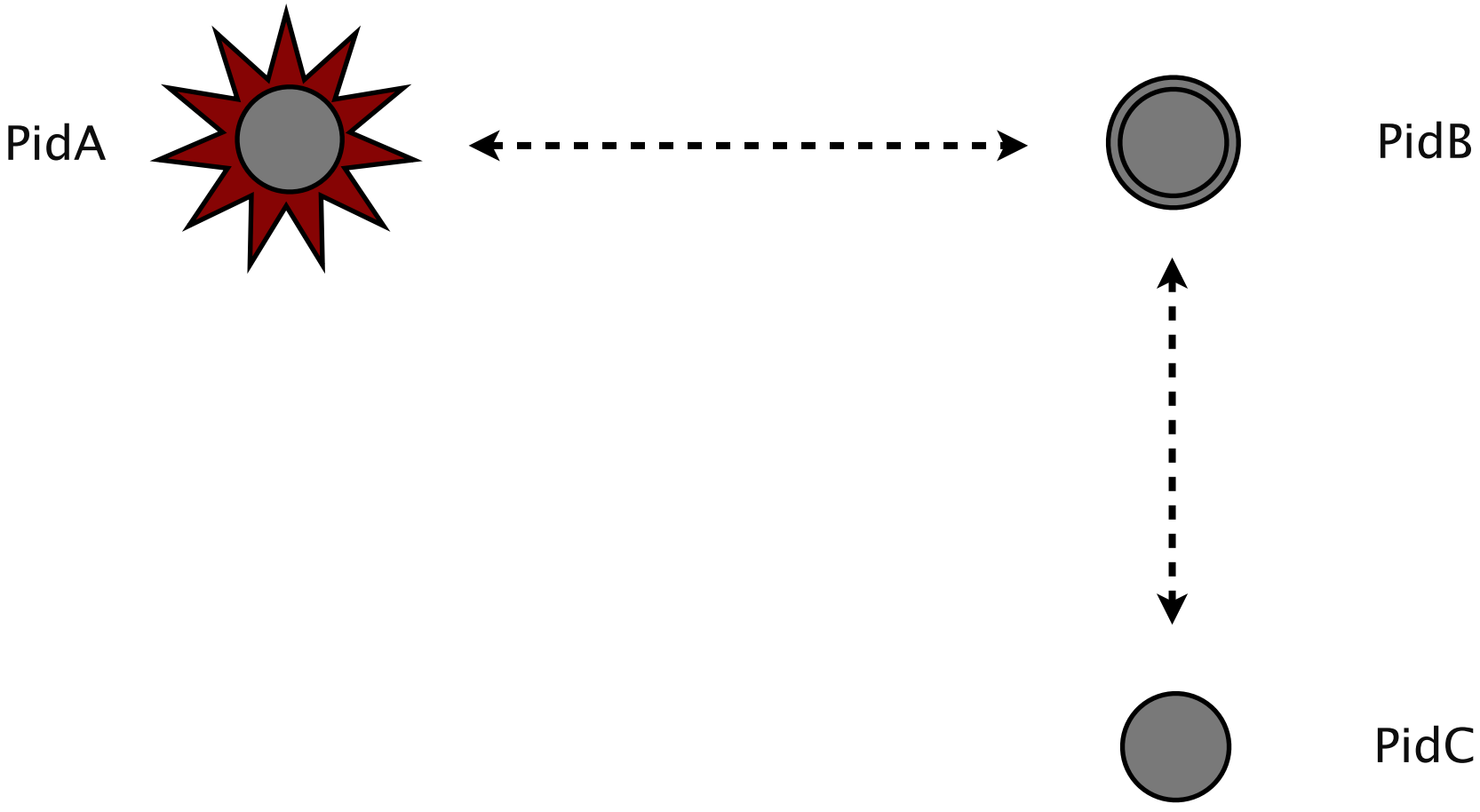
Propagating Exit Signals

Erlang SOLUTIONS

Trapping Exits

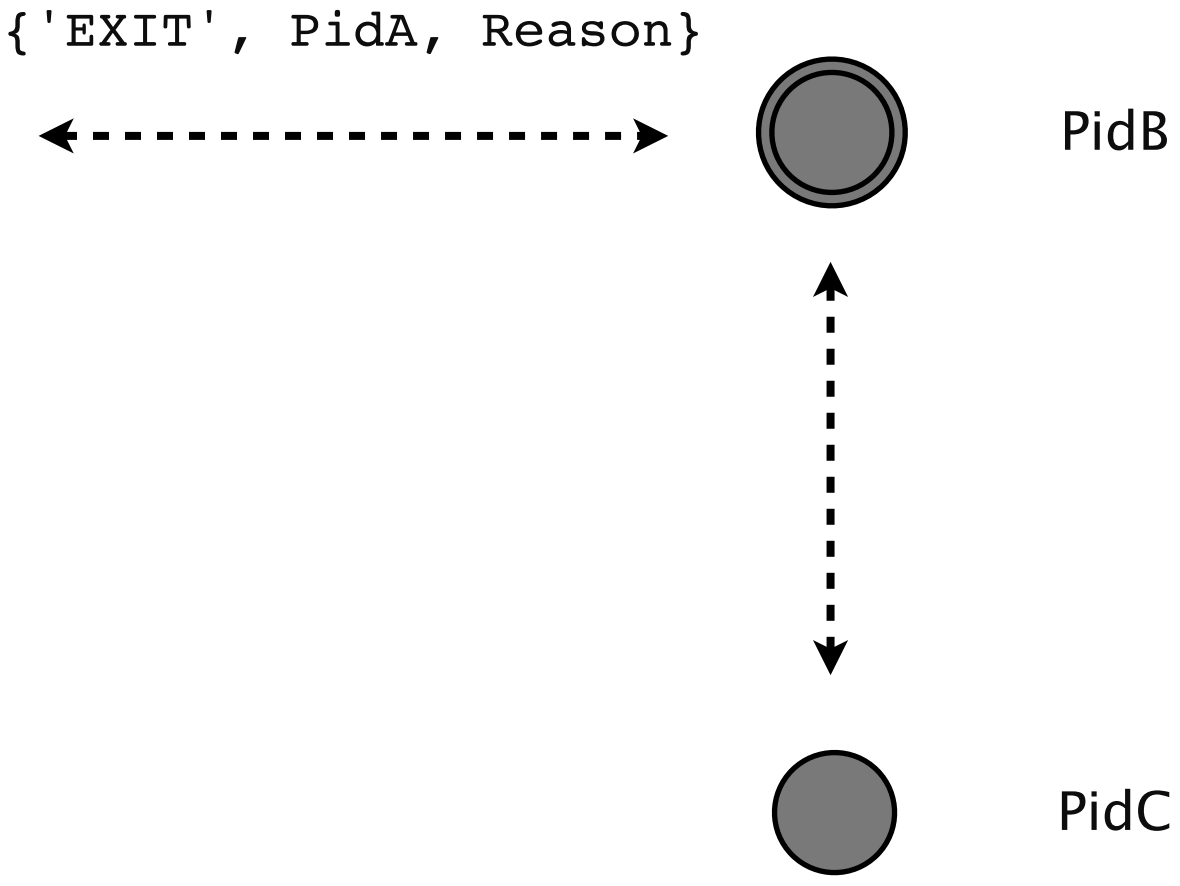


Trapping Exits

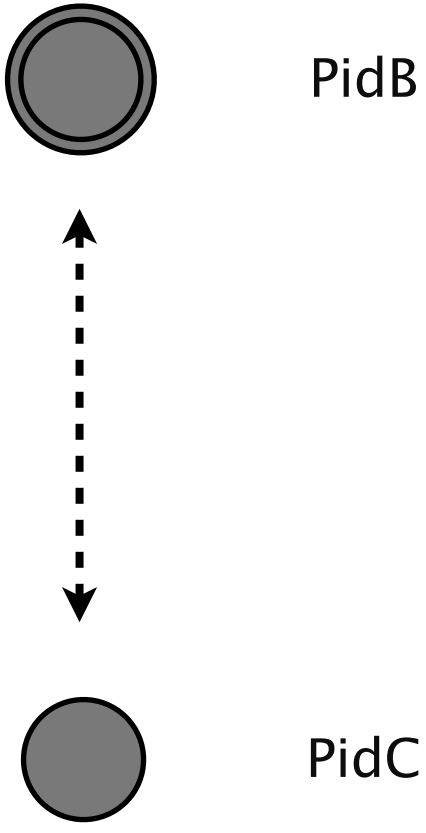


Erlang SOLUTIONS

Trapping Exits

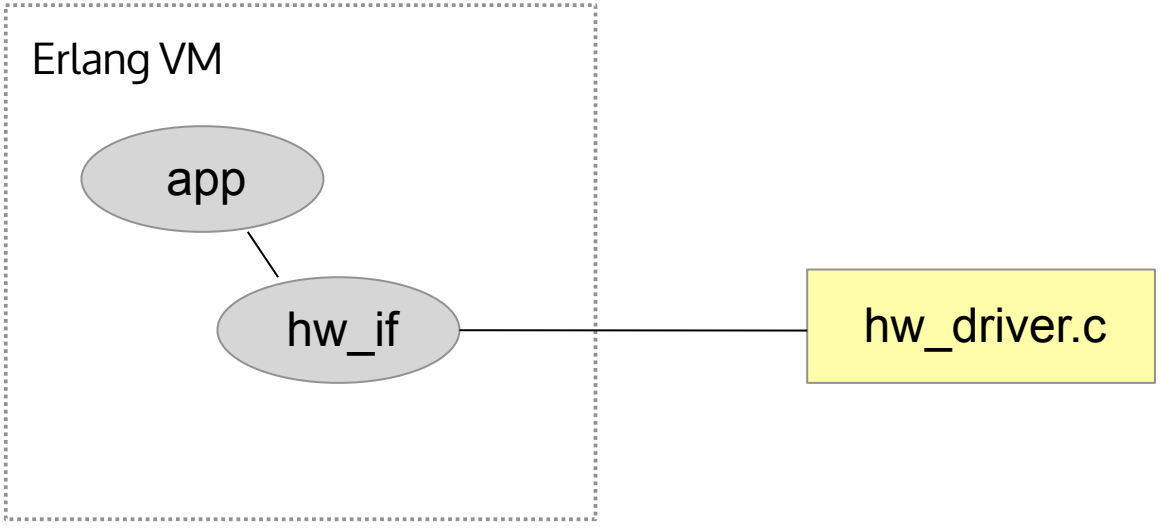


Trapping Exits



External Interfaces

- Native Implemented Functions (NIFs) and ports used to interface external world to the Erlang runtime.



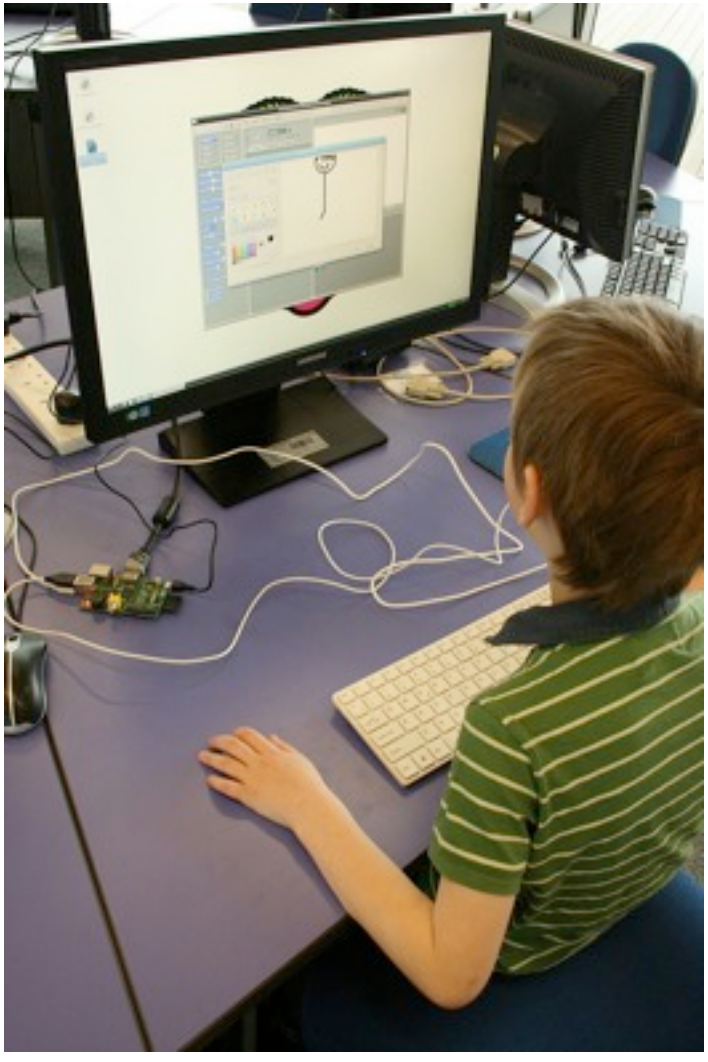
Erlang, the Maestro

(flickr/dereckesanches)

Raspberry Pi

- 700 MHz ARM11
- 256 MB DDR2 RAM
- 10/100Mb Ethernet
- 2x USB 2.0
- (HDMI, Composite Video, 3.5mm Stereo Jack, DSI, CSI-2)

Raspberry Pi in Education



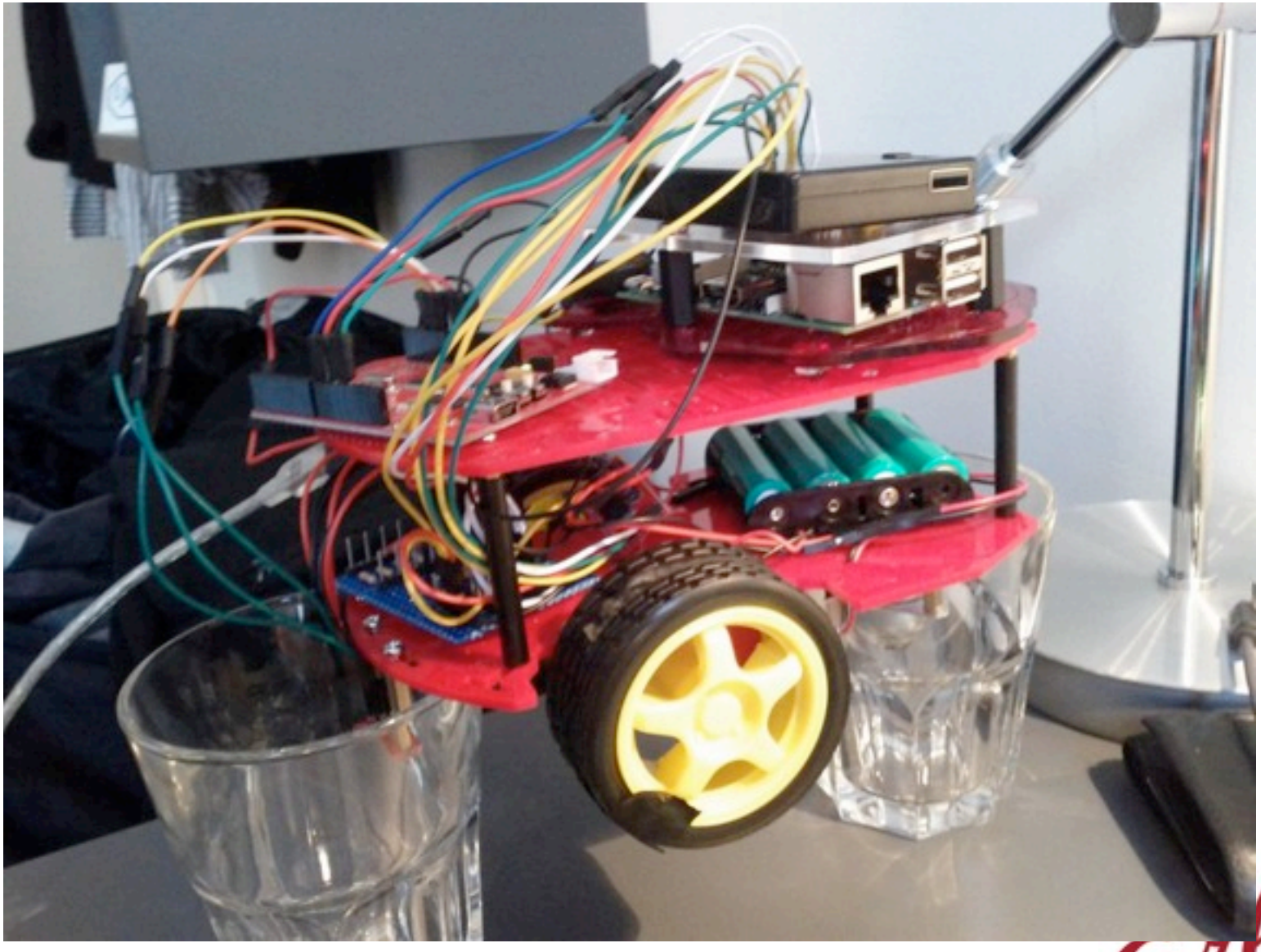
(flickr/lebeus)

- The Raspberry Pi Foundation is a UK registered charity.
- Mission statement: "...to promote the study of computer science and related topics, especially at school level, and to put the fun back into learning computing."
- Future Engineers/Programmers!

Raspberry Pi Peripherals

- GPIO
- UART
- I2C
- I2S
- SPI
- PWM
- DSI
- CSI-2

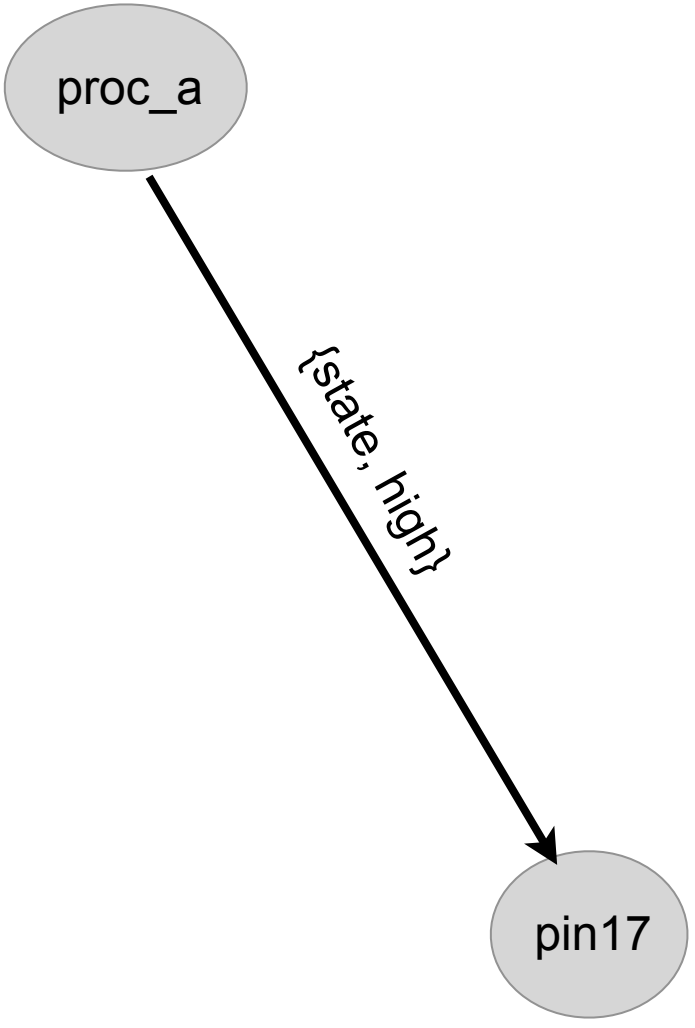
The ErlBuggy!



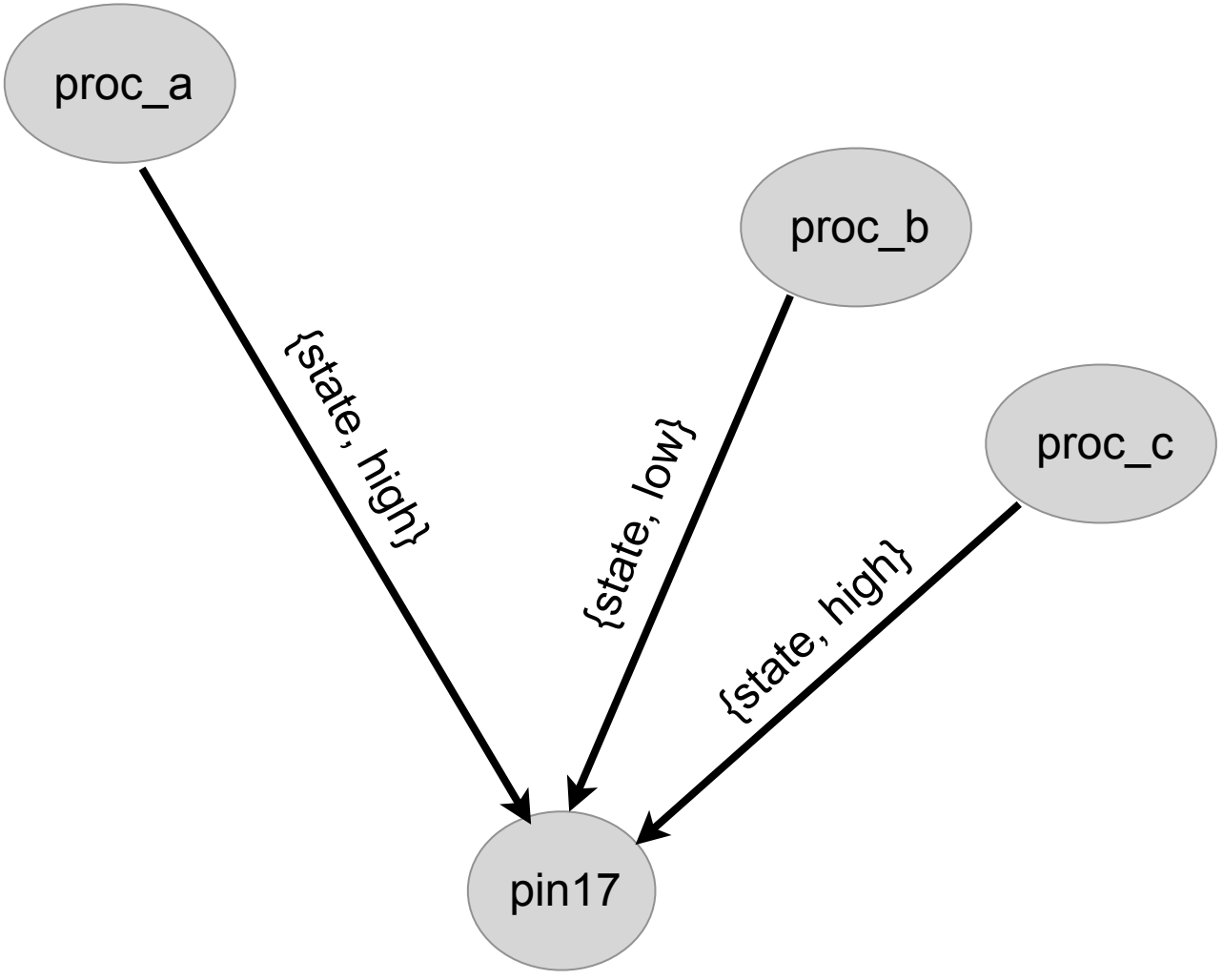
ErlBuggy

http://vimeo.com/48375416

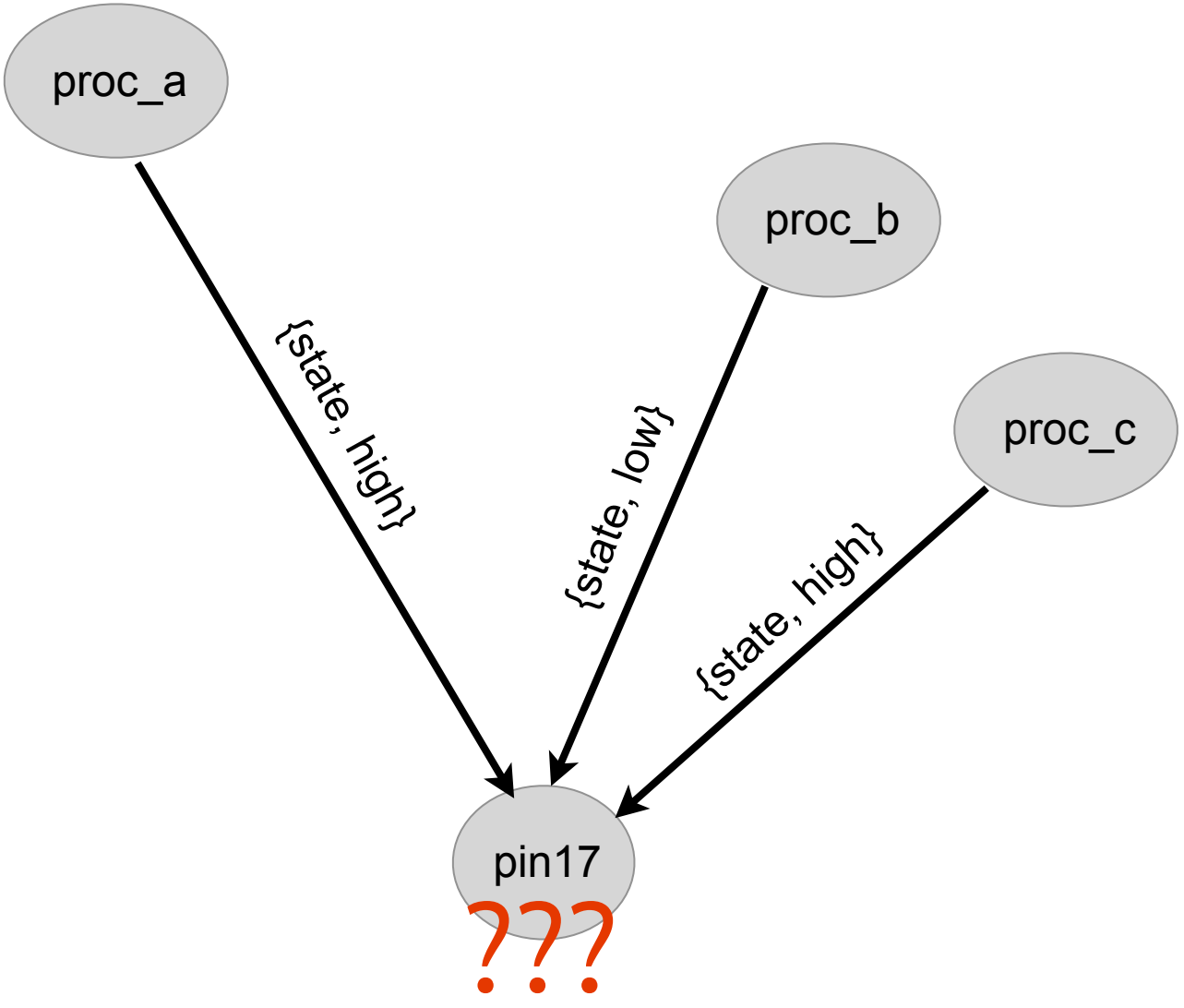
Example: GPIO



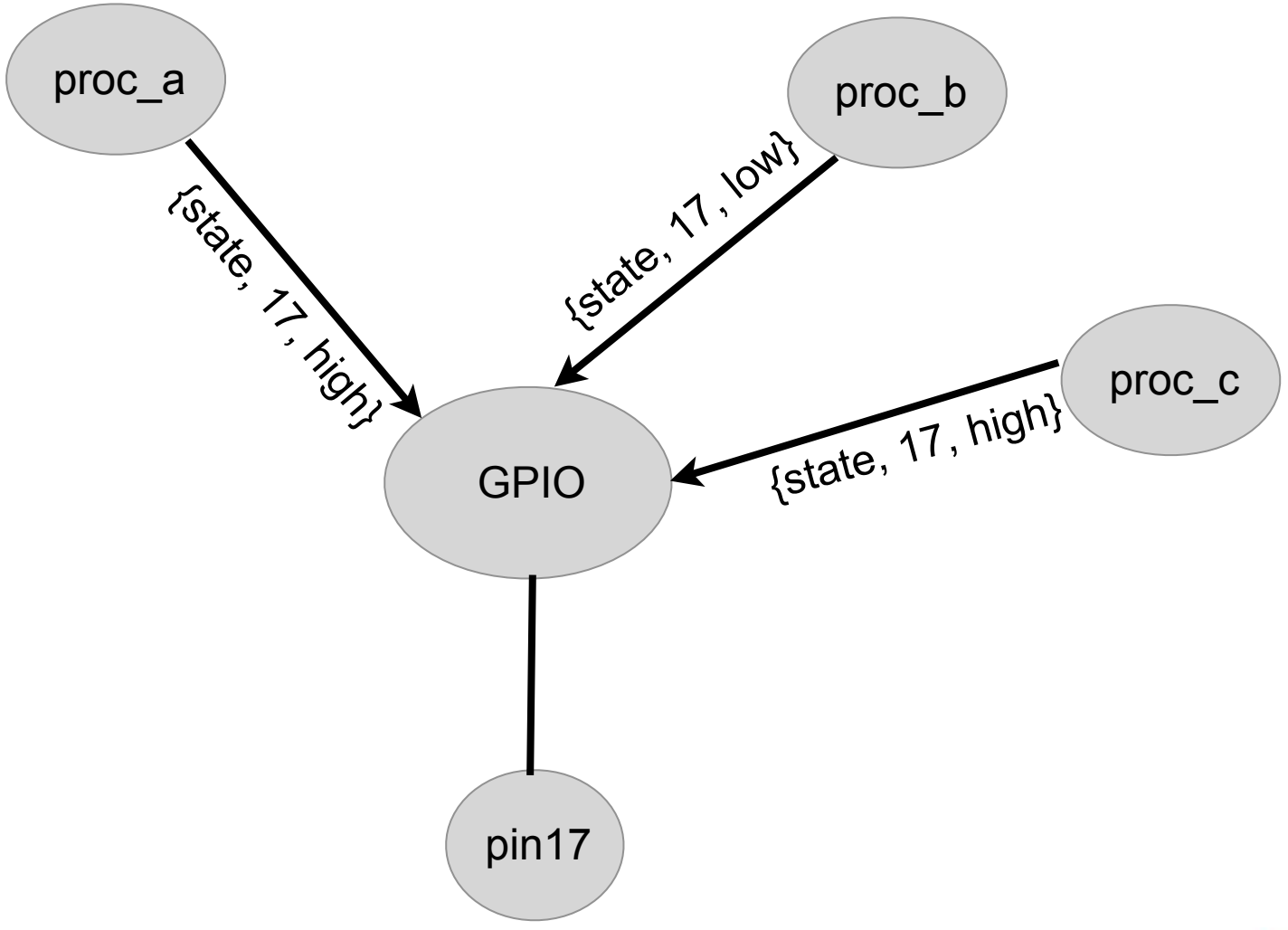
Example: GPIO



Example: GPIO



Example: GPIO



GPIO Proxy

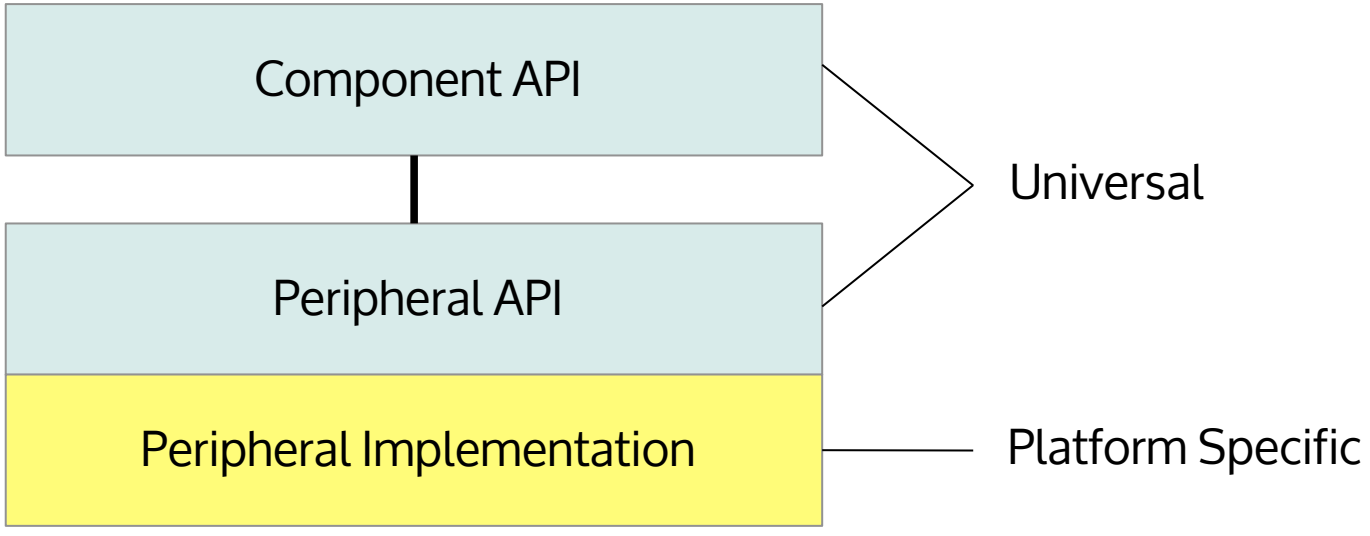
- Replaces ‘locks’ in traditional sense of embedded design
  - Access control/mutual exclusion
- Can be used to implement safety constraints
  - Toggling rate, sequence detection, direction control, etc.

Erlang SOLUTIONS

Fine Grain Abstraction

- Advantages
  - Application code becomes simpler
  - Concise and shorter modules
  - Testing becomes easier
  - Code re-use (potentially) increases
- Disadvantage
  - Architecting fine grain systems is difficult

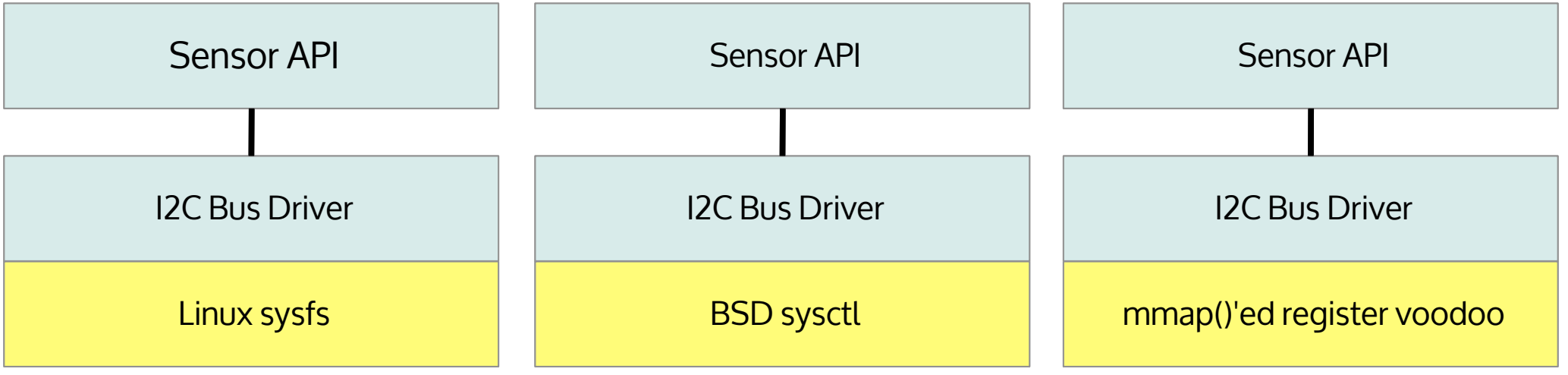
Universal Peripheral/Component Modules



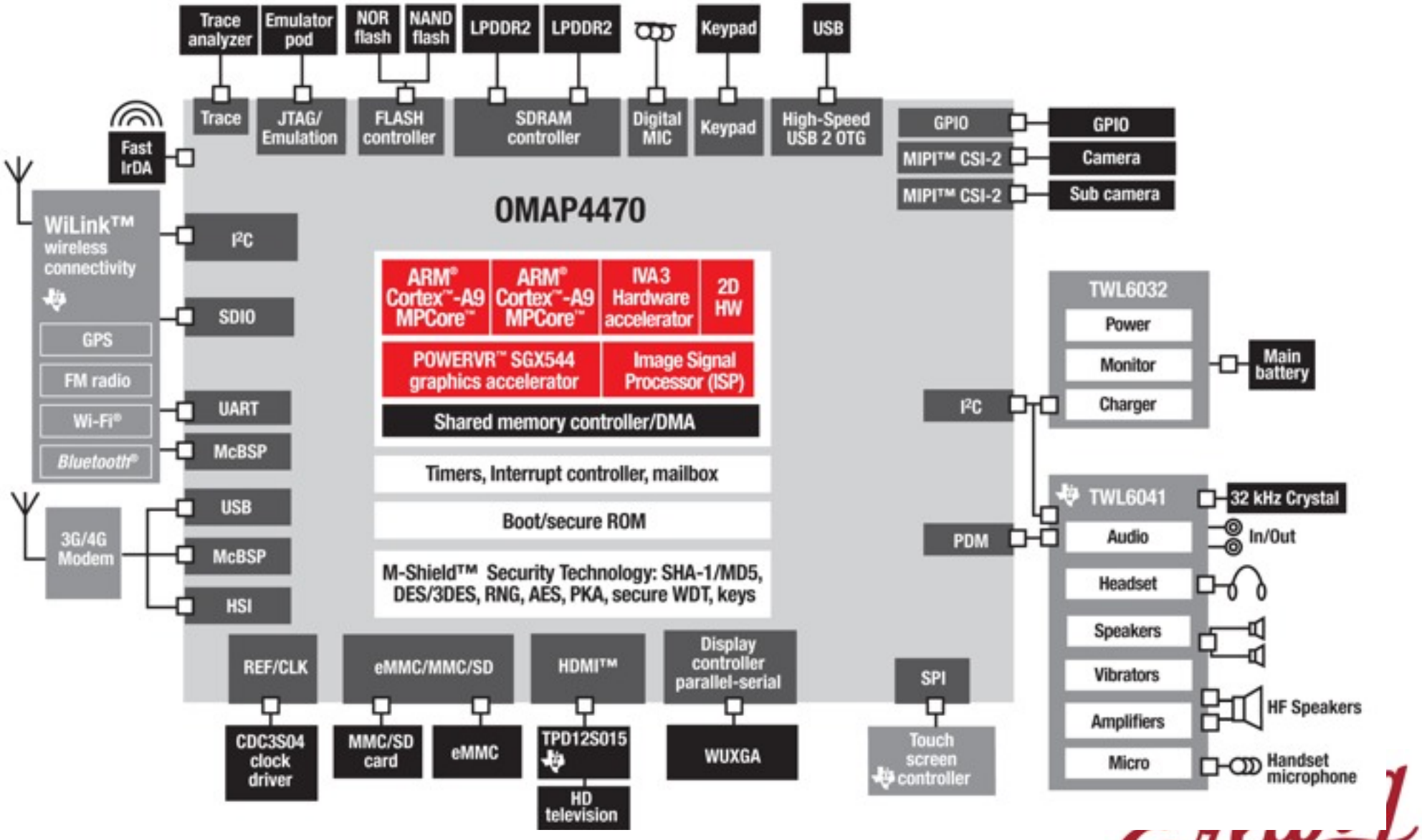
Universal Peripheral/Component Modules

An Example:

Temperature Sensor with I2C Interface



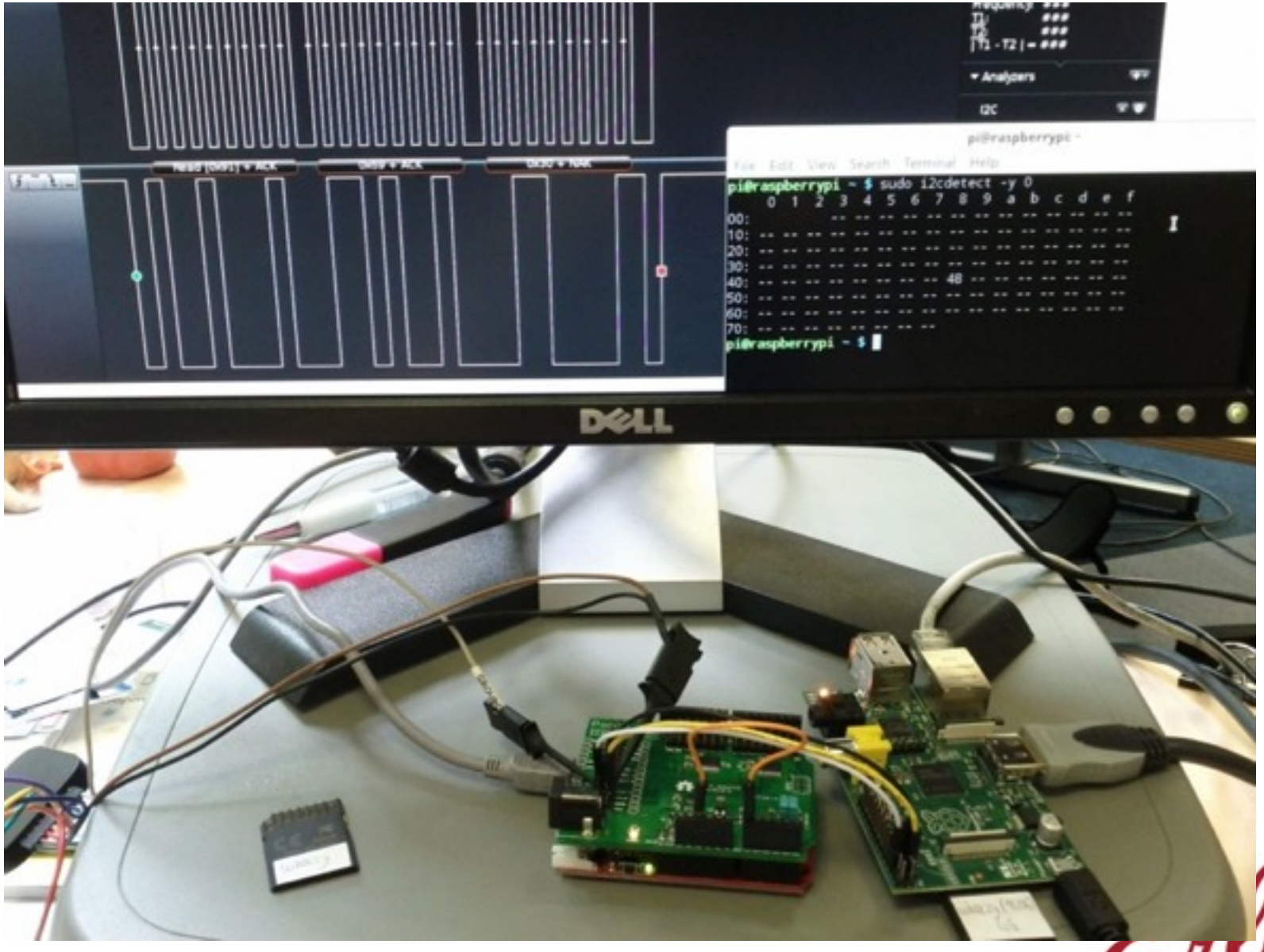
TI OMAP Reference System



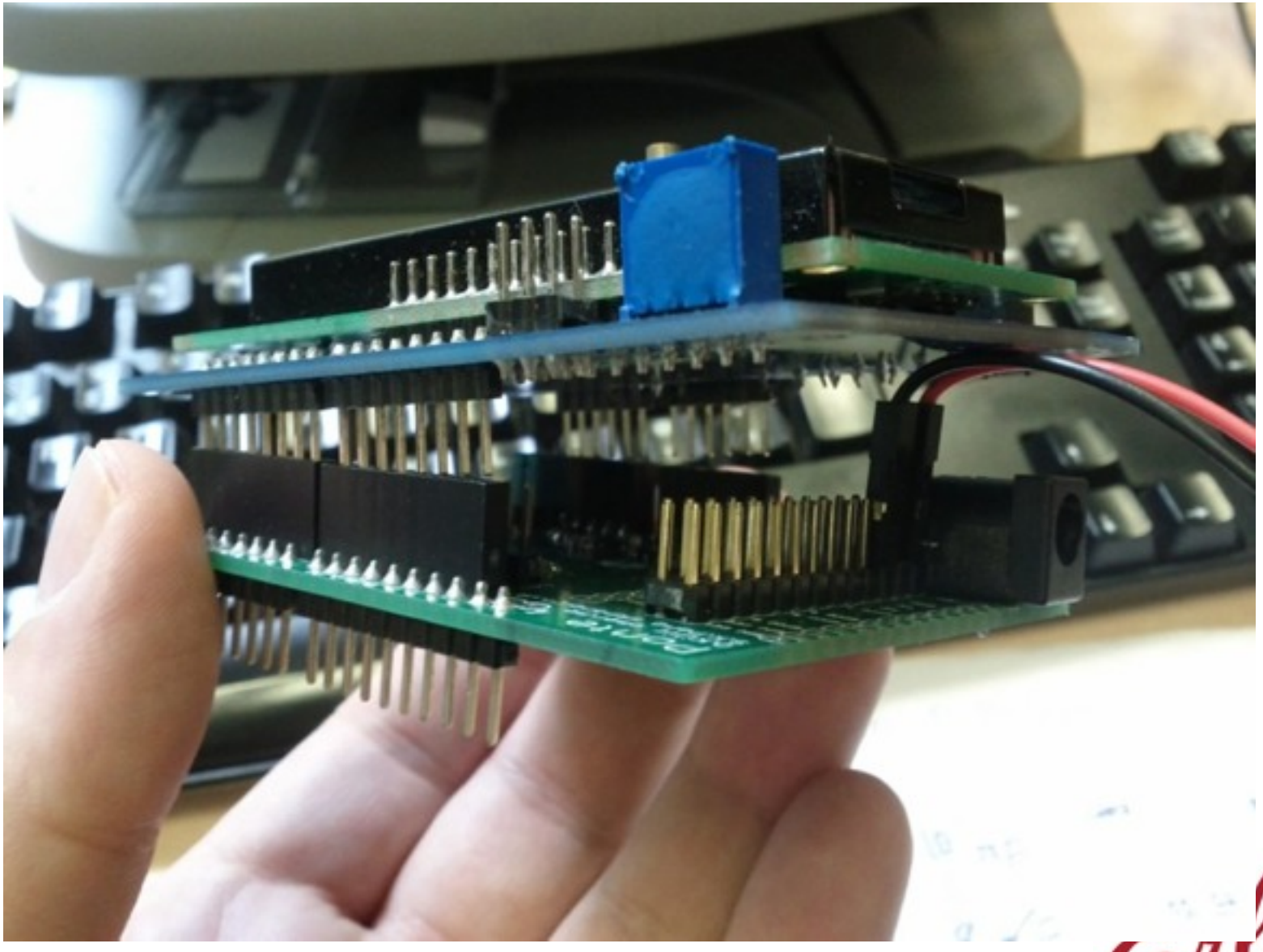
Ponte Prototype

(flickr/carrierdetect)

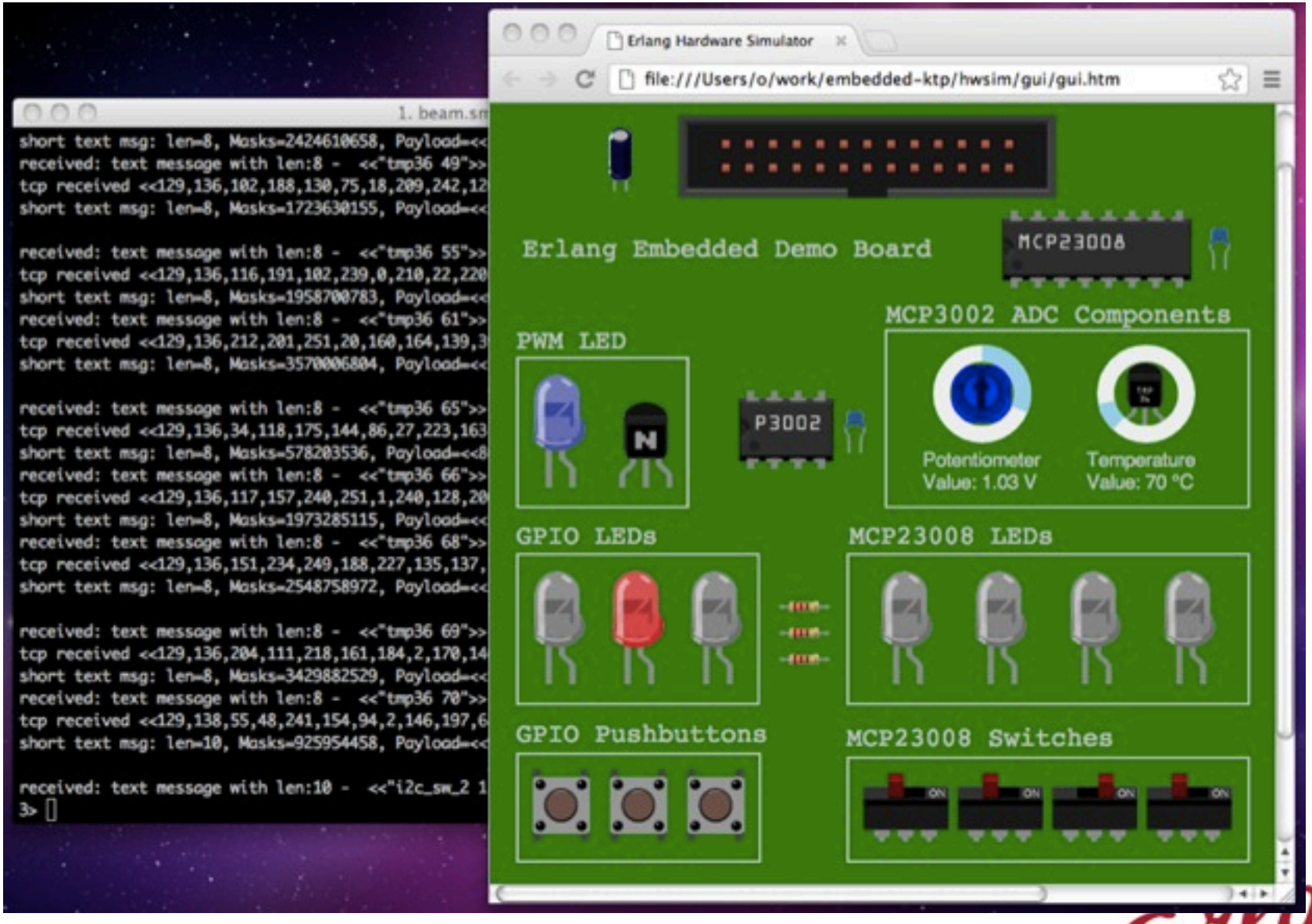
Ponte Prototype



Ponte Prototype

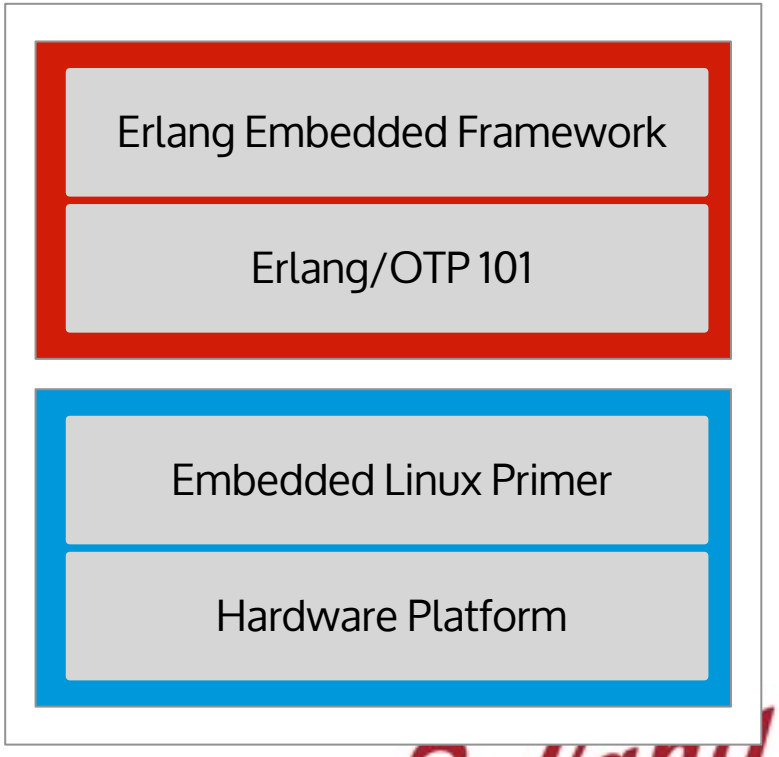


Hardware Simulator (WIP)



Erlang Embedded Training Stack

- A complete package for people interested in developing the next generation of concurrent/ distributed Embedded Systems
- Training Modules
  - Embedded Linux Primer
  - Erlang/OTP 101
  - Erlang Embedded Framework



Get in touch if you're interested.

Erlang SOLUTIONS

Thank you

- http://erlang-embedded.com
- embedded@erlang-solutions.com
- @ErlangEmbedded

> Any sufficiently complicated concurrent program in another language contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of Erlang.

– Robert Virding
Co-Inventor of Erlang