Basic JITPCB

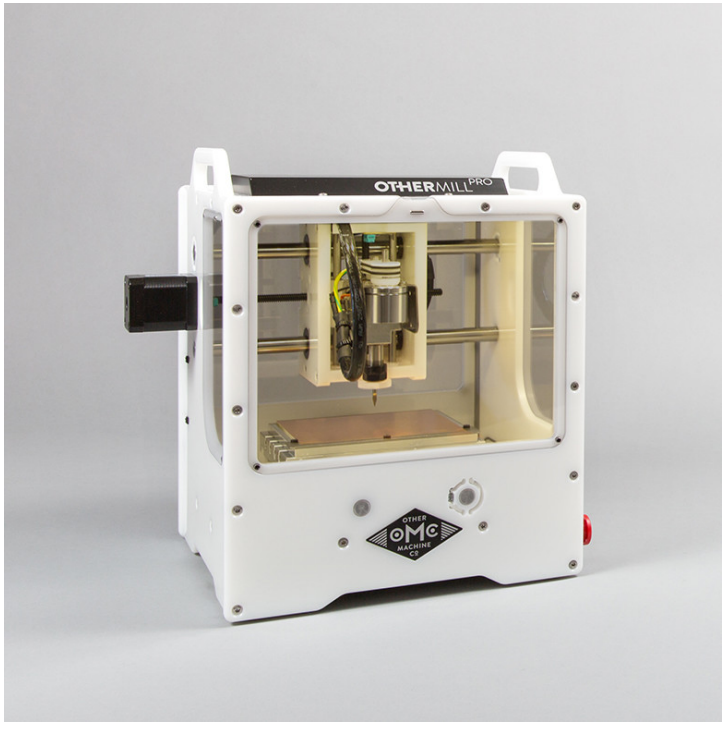
Jonathan Bachrach

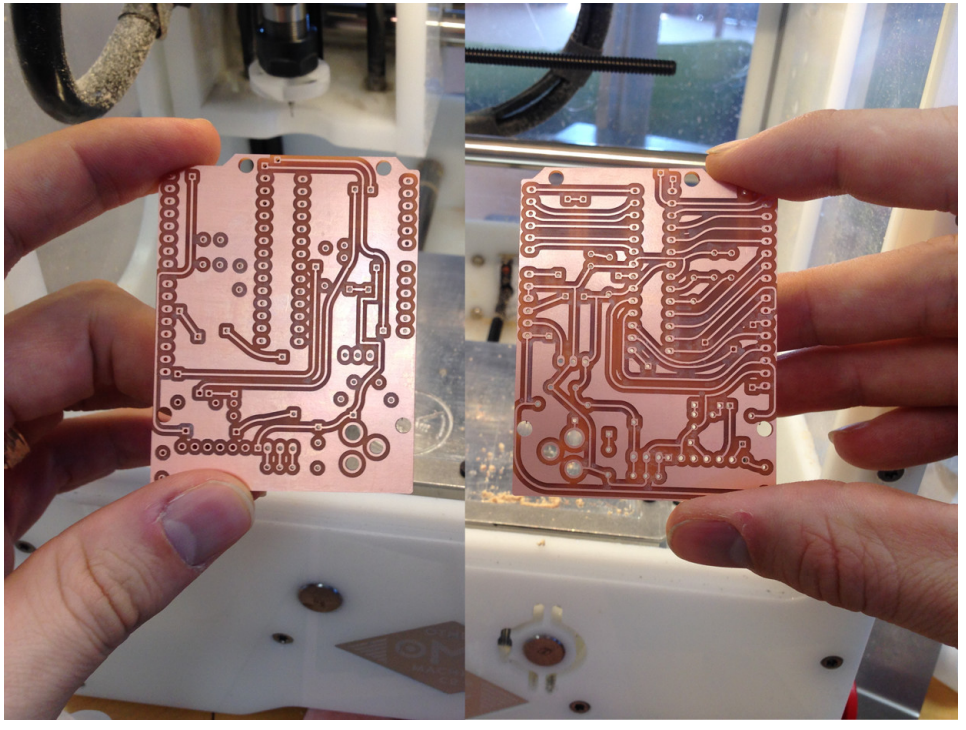
EECS UC Berkeley

September 15, 2016

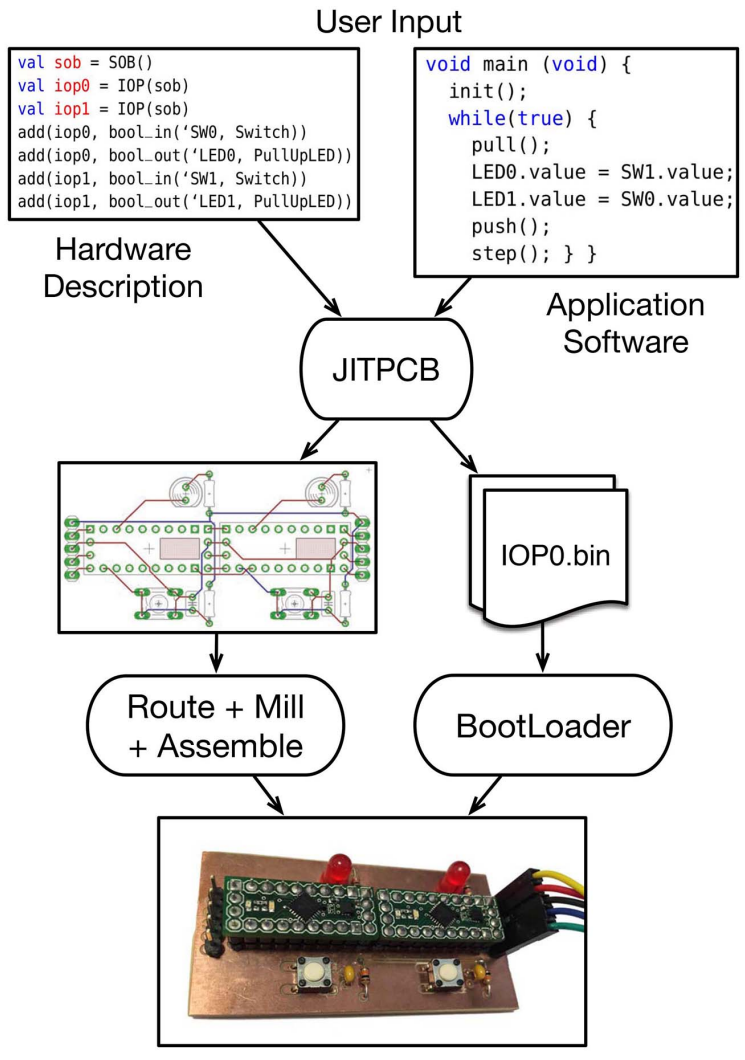
Last Time

- Milling PCBs





- Basic JITPCB



JITPCB Abstract

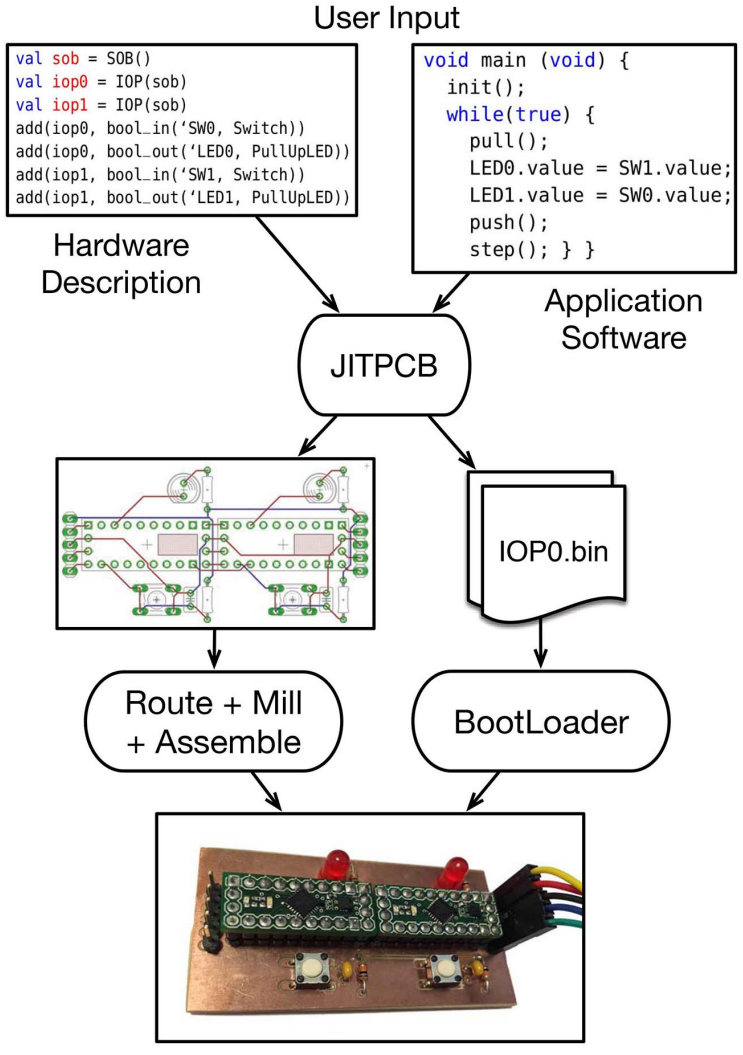
Declaratively design embedded systems

Inputs

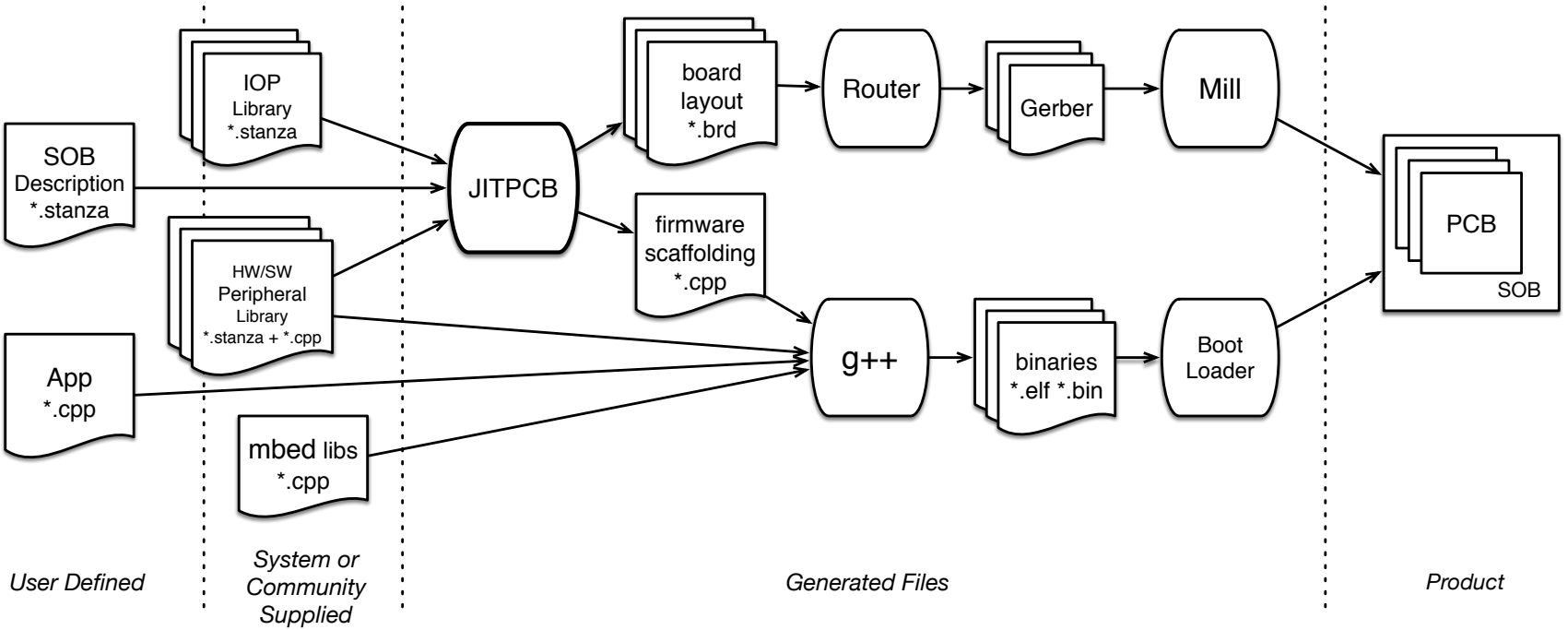
- list of peripherals
- application in C++

Outputs

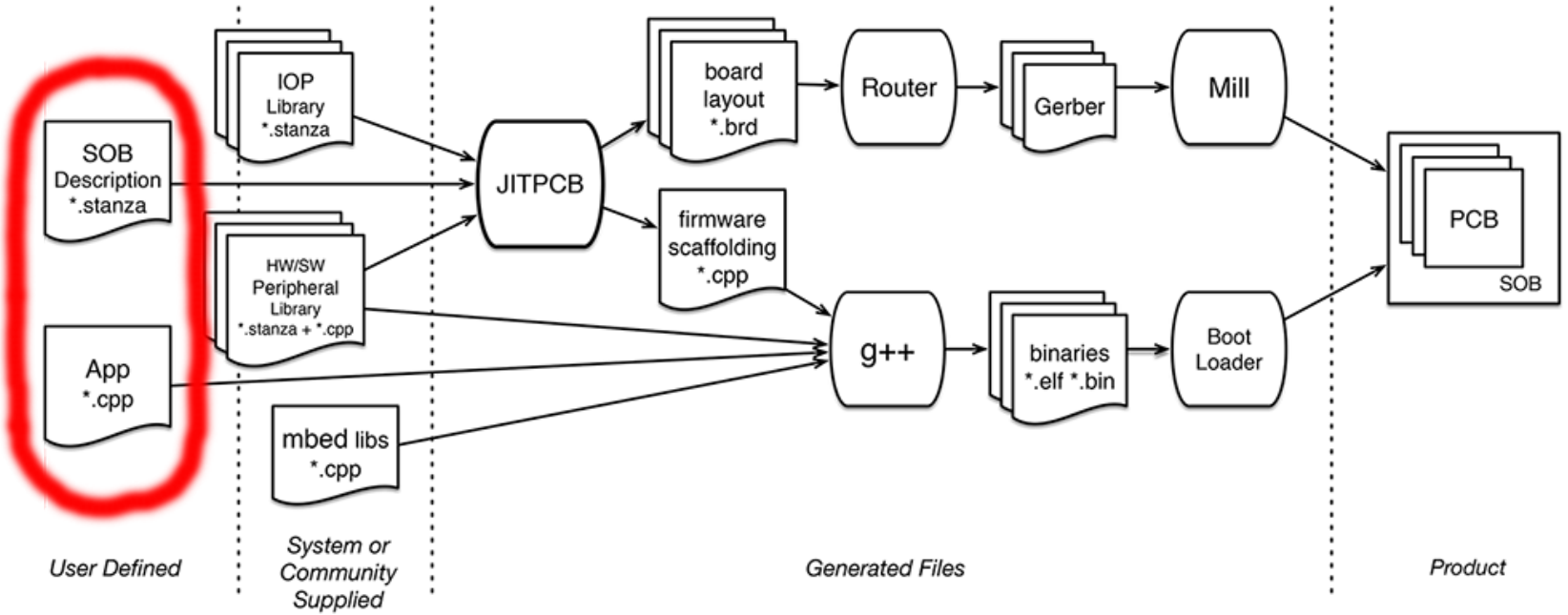
- (multiple) board files
- compiled firmware
- networking layer



Entire JITPCB Workflow

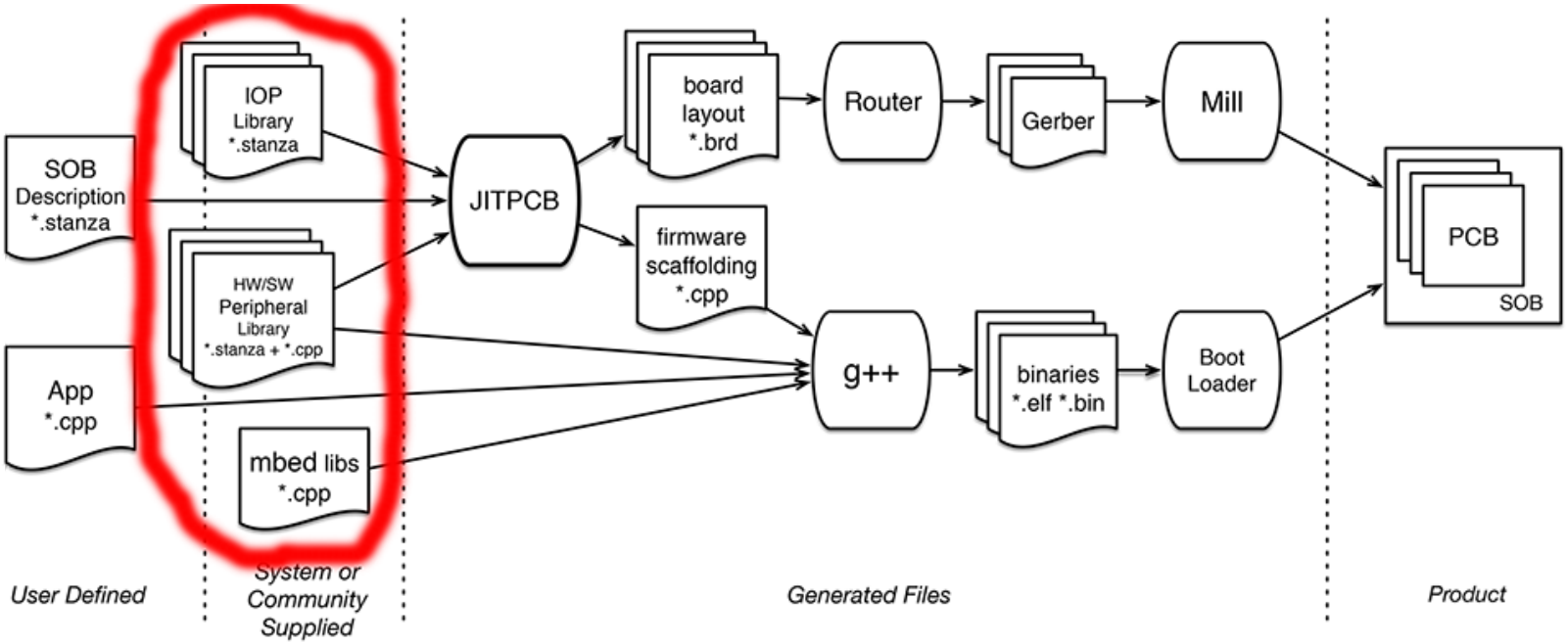


- SOB description without layout
- App program



Expert JITPCB

- SOB description with layout
- Peripherals
- Components
- CPUs

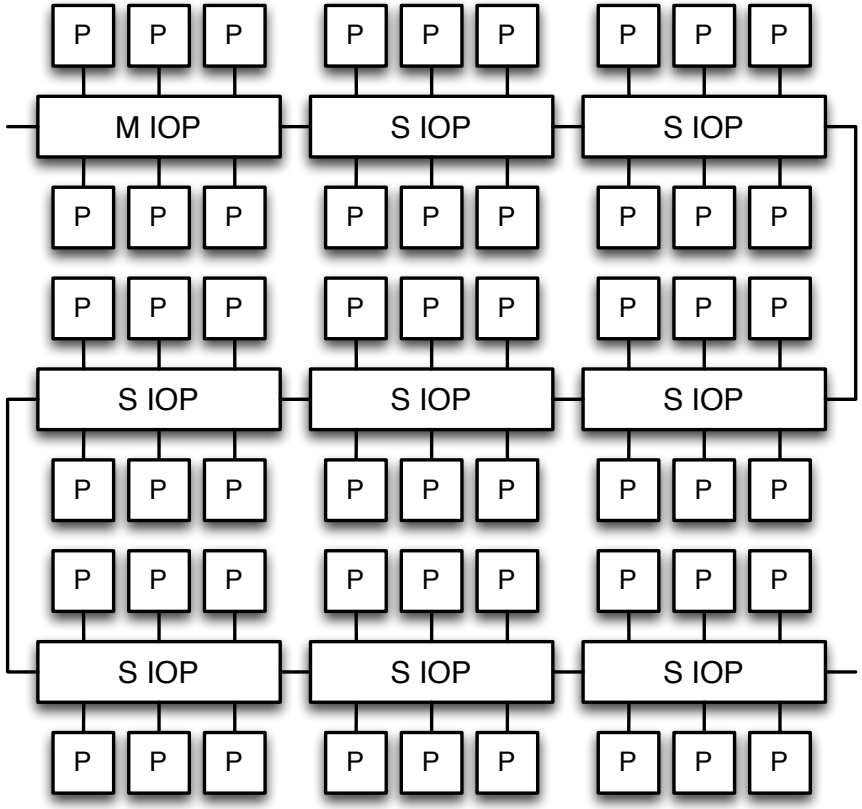


Template

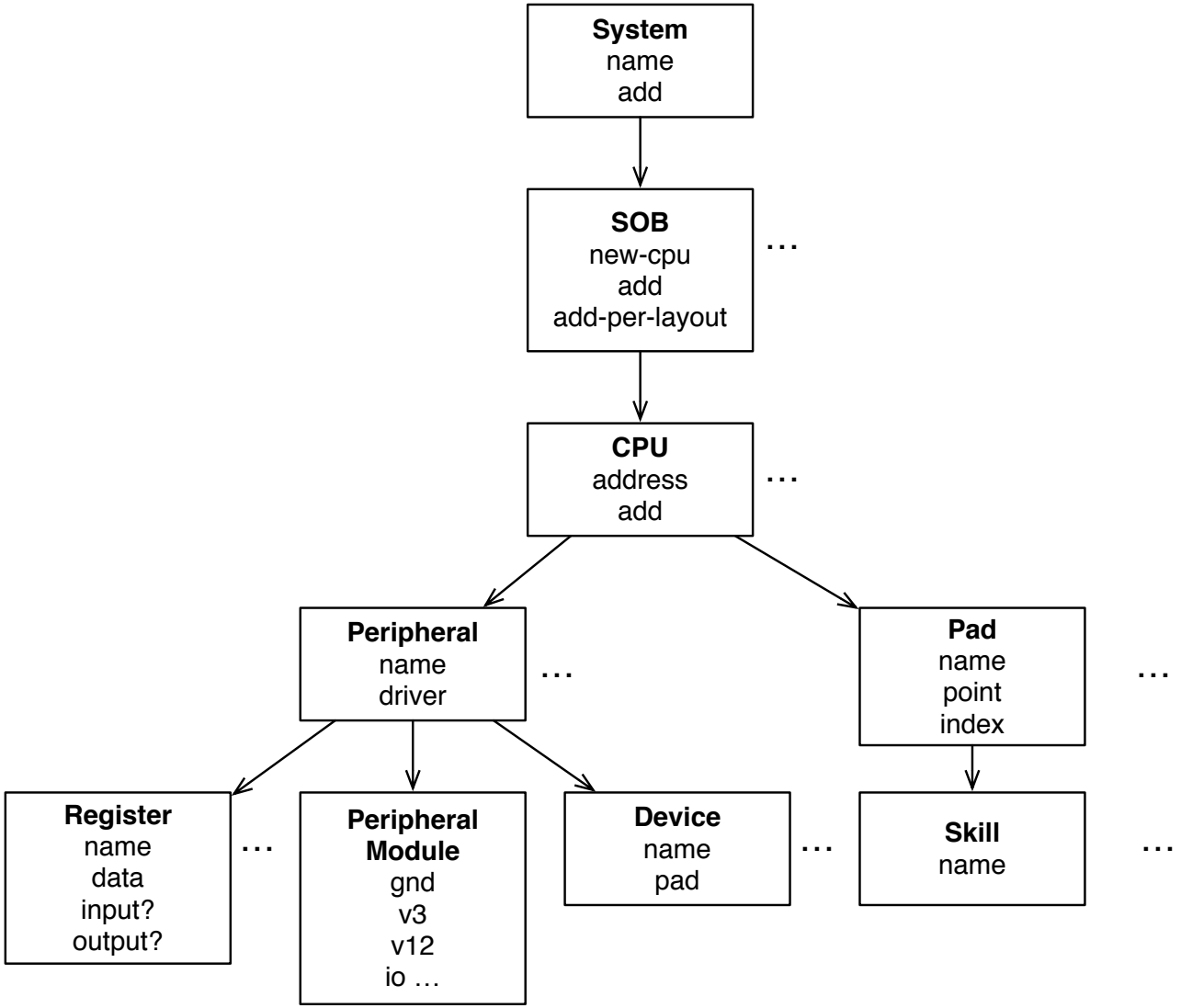
- Peripherals
- IO processors
  - one master and rest slaves
  - peripherals attach to pins
- Network

Benefits

- Modular and Scalable
- Easier SW and Routing



- system
- sob
- cpu
- peripherals



Hello World App


```
defn hello-world () :
  val sys = System(‘hello-world)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, bool-out-peripheral(‘led0, LEDio))
  add(sob, bool-out-peripheral(‘led1, LEDio))
  add(sob, bool-out-peripheral(‘led2, LEDio))
  add(sob, bool-in-peripheral(‘button, SWio))
  sys

defn main () :
  reg-librarian(Librarian())
  gen(hello-world())
```

- system of system of boards (SOSOB)

```
public deftype System
public defn System (name:Symbol) -> System ...
public defmulti name (sys:System) -> Symbol
public defmulti add (sys:System, elt:SOB) -> SOB
public defmulti sobs (s:System) -> Vector<SOB>
public defn gen (sys:System) ...
```

- CPUs organized in grid and connected by I2C
- one CPU is designated master

```
public deftype SOB
public defn SOB (cons-rep: () -> CPUrep, have-master?:True|False) -> SOB ...
public defmulti add (s:SOB, p:Peripheral)
```

- model of CPU
- one per CPU used in system
- describes pads and devices on CPU

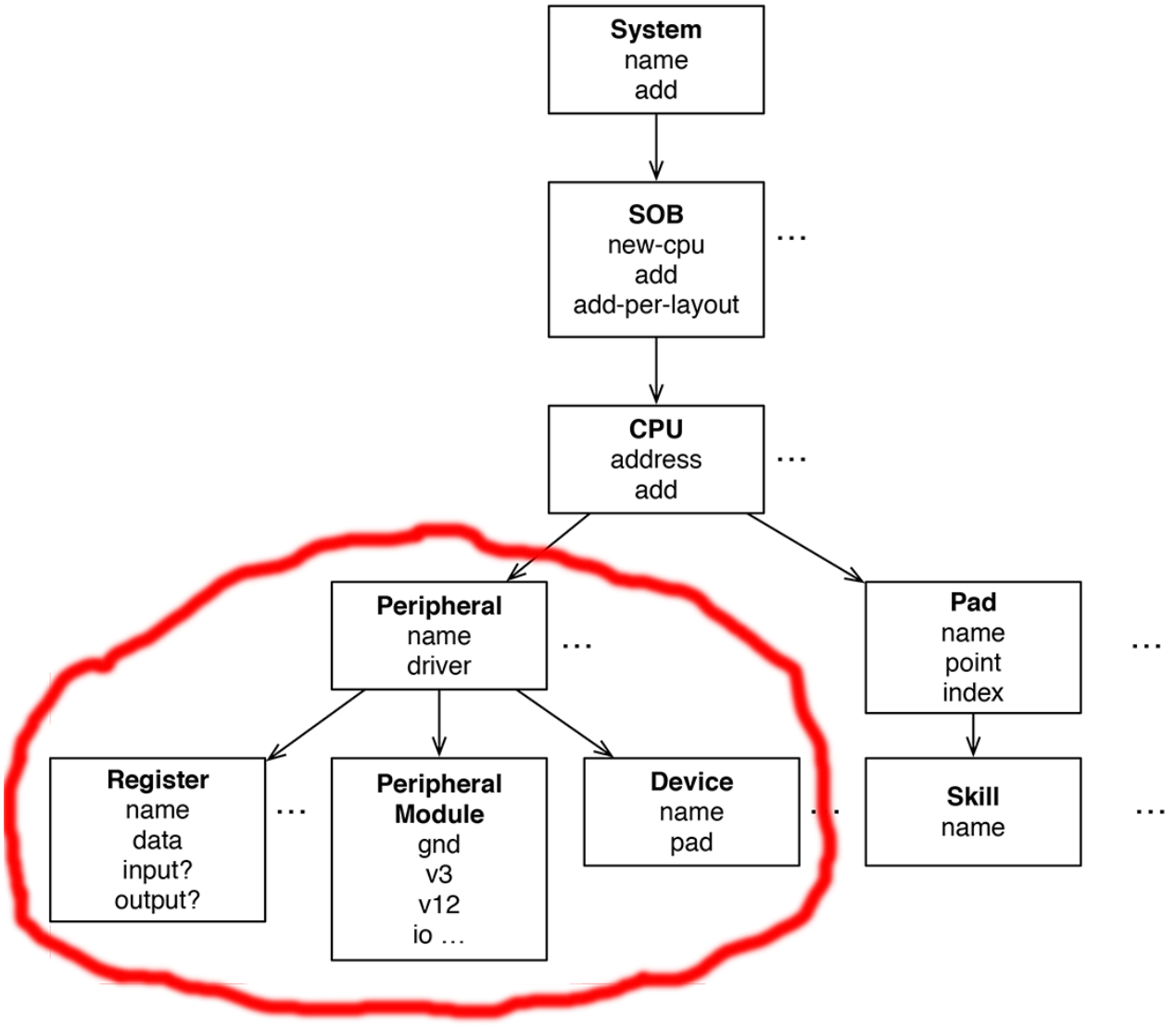
```
public defstruct CPUrep :
  pads : Seqable<Pad>
  fab : (Symbol, String) -> CPU_PKG
  sda-pin-name : Symbol
  scl-pin-name : Symbol
```

- connection to real port on cpu
- device skills

```
public deftype Pad<T> <: Equalable
public defmulti skills<?T> (p:Pad<?T>) -> Seqable<Symbol>
...
```

Meta Modeling of Peripherals

- peripherals



Peripheral

- describes user peripheral on board
- groups sw driver and hw manifestation

```
public deftype Peripheral
```

```
public defn bool-out-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) -> Peripheral ...
public defn bool-in-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) -> Peripheral ...
public defn pwm-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) -> Peripheral ...
```

Peripheral Definition

- devices needed by cpu
- registers used by driver

```
public defn bool-in-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) :
  Peripheral(name, "BoolIn", hw,
             [Device(‘DigitalIn, pad#io)],
             [Register(‘value, CharType("0"), true, false)])
```

```
public defn bool-out-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) :
  Peripheral(name, "BoolOut", hw,
             [Device(‘DigitalOut, pad#io)],
             [Register(‘value, CharType("0"), false, true)])
```

```
public defn pwm-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) :
  Peripheral(name, "PWM", hw,
             [Device(‘PwmOut, pad#io)],
             [Register(‘value, FloatType("0.0"), false, true)])
```

- spec name and hardware

```
public defn bool-in-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn bool-out-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn bool-in-out-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn analog-in-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn pwm-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn uart-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
```

- register models peripheral driver control registers
- device models cpu peripherals that peripherals need

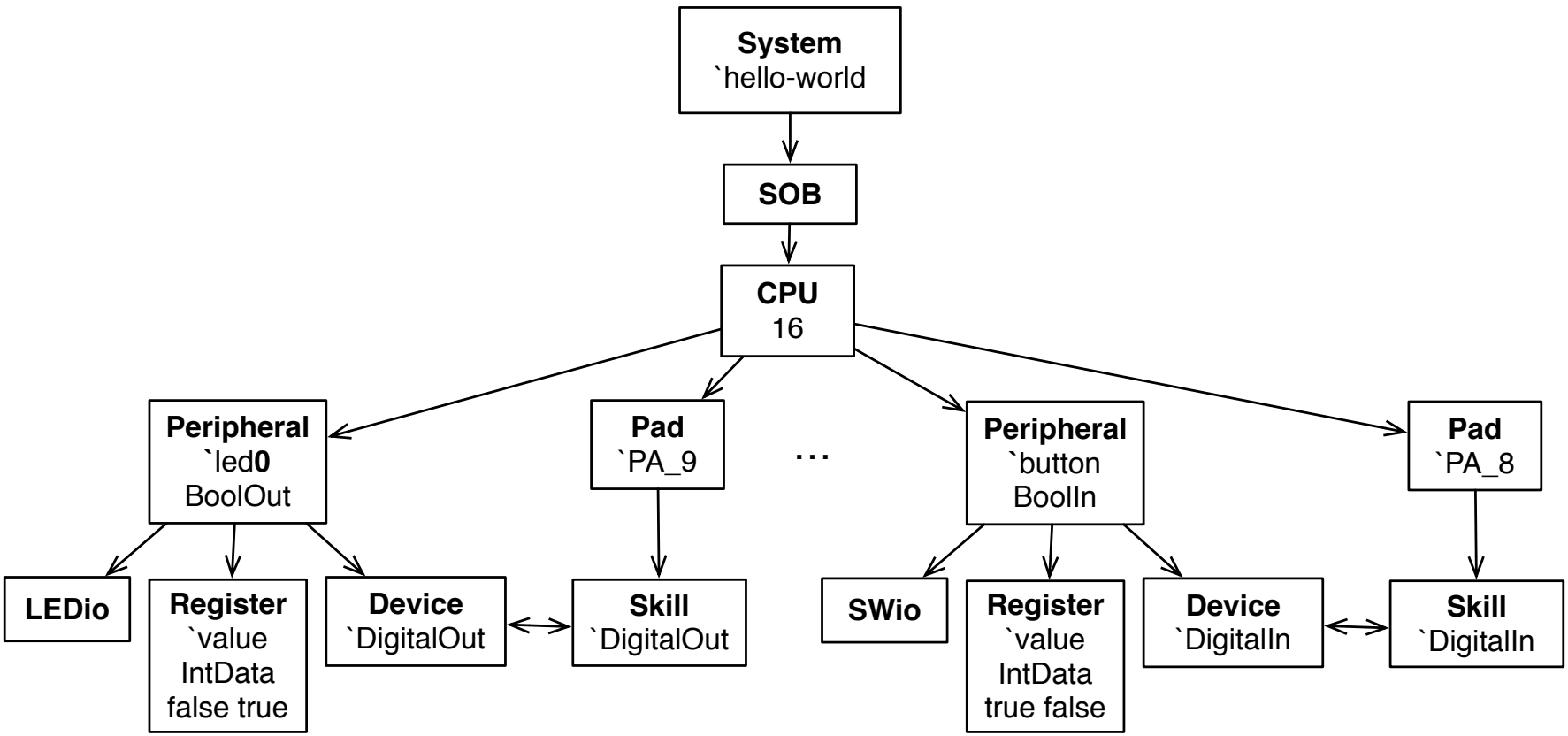
```
public deftype Register
public defn Register (name:Symbol, data:Data,
                      input?:True|False, output?:True|False) -> Register
```

```
public deftype Data
public defmulti size (d:Data) -> Int
public defmulti type (d:Data) -> String
public defmulti init (d:Data) -> String
public defn Data (size:Int, type:String, init:String) -> Data ...
```

```
public deftype Device
public defn Device (name:Symbol, to-ref: (?) -> AnySignal) -> Device
```

Hello World Model


```
defn hello-world () :
  val sys = System(‘hello-world)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, bool-out-peripheral(‘led0, LEDio))
  ...
  add(sob, bool-in-peripheral(‘button, SWio))
  sys
```



- hardware organized into hierarchy of anymodules
- components are packages with pads

```
public deftype Module <: AnyModule
public deftype Component <: AnyModule
```

```
public deftype AnySignal
public deftype Ref <: AnySignal
public deftype Signal <: AnySignal
```

```
public deftype CPU_PKG <: AnyModule
```

```
public deftype PeripheralModule <: Module
```

Component Descriptions:

```
defcomponent LED ("LED", "LED5MM") :
  pad A
  pad K

defcomponent Resistor ("RCL", "0204/7") :
  pad 1
  pad 2
```

Module Description:

```
defmodule LEDio () <: PeripheralModule:
  ;; interface
  inherit gnd
  inherit v3
  sig io
  ;; subcomponents
  mod l   : LED("red")
  mod r   : Resistor("1k")
  ;; netlists
  sig up  = [r.1, l.A]
  sig v3  = [r.2]
  sig gnd = []
  sig io  = [l.K]
  ;; layout
  lay HBox([Rot(r,90), Rot(led,90)], 1.0)
```

Peripheral Definition

- devices needed by cpu
- registers used by driver

```
public deftype PeripheralModule <: Module
public defmulti pad#gnd (m:PeripheralModule) -> Signal
public defmulti pad#v12 (m:PeripheralModule) -> Signal
public defmulti pad#v5 (m:PeripheralModule) -> Signal
public defmulti pad#v3 (m:PeripheralModule) -> Signal
public defmulti pad#io (m:PeripheralModule) -> Signal
```

- used to arrange packages as in Eagle
- must set a librarian up before creating

```
public deftype Library
```

```
public deftype Librarian
public defn Librarian () -> Librarian ...
public defn reg-librarian (libr:Librarian) ...
```

```
defn main () :
  reg-librarian(Librarian())
  gen(hello-world())
```

- virtualized peripherals – RPC calls
- bootloader

hello-world.cpp

- example app has main
- peripherals are wrapped
- access much like in mbed
- of course logic agnostic to peripheral placement
- defines `iop` and need to call `init` and `step`

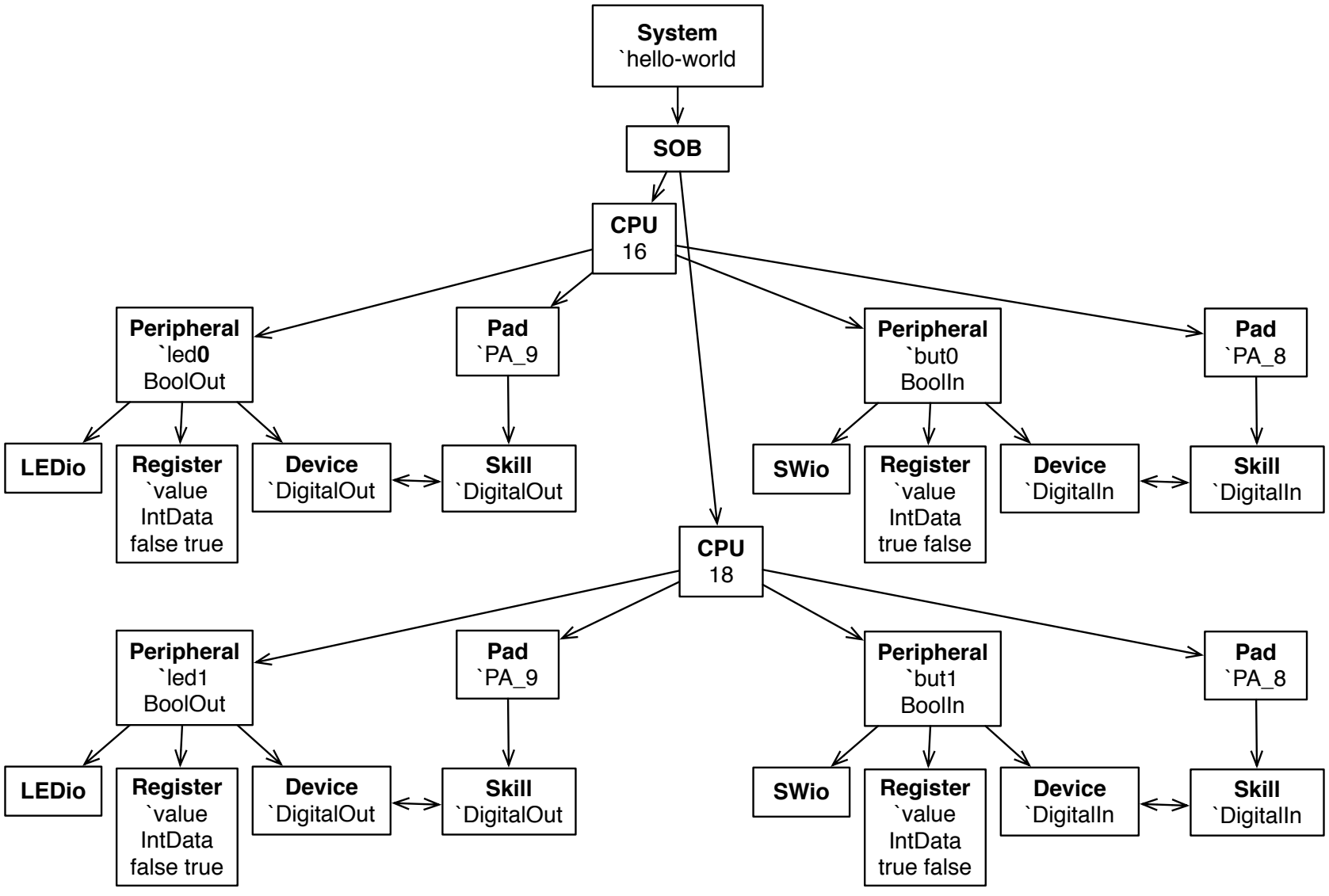
```
#include "master.h"

int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    led0.value = button.value;
    led1.value = !button.value;
    led2.value = button.value;
    iop.step();
  }
}
```

- multiple cpus created explicitly with `new-cpu`

```
defn dual-hello-world () :
  val sys = System(‘hello-world)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  val cpu0 = new-cpu(sob, true)
  val cpu1 = new-cpu(sob, false)
  add(sys, sob)
  add(cpu0, bool-out-peripheral(‘led0, AnyLEDio{_, LED-DIP, Res}))
  add(cpu0, bool-in-peripheral(‘but0, SWio))
  add(cpu1, bool-out-peripheral(‘led1, AnyLEDio{_, LED-DIP, Res}))
  add(cpu1, bool-in-peripheral(‘but1, SWio))
  sys
```

Hello World Model



dual-hello-world.cpp

- write code as if peripherals are on same cpu

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    led0.value = but1.value;
    led1.value = but0.value;
    iop.step();
  }
}
```

Arduino Boards with JITPCB

- physically sound
- automatic pin assignment
- scalable

- rgb-led peripheral
- photocell peripheral

```
defn arduino-color-mixing-lamp () :
  val sys = System(‘arduino-color-mixing-lamp)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, rgbled-peripheral(‘rgbled, RGBLEDio, true))
  add(sob, analog-in-peripheral(‘photo_r,PhotocellIO))
  add(sob, analog-in-peripheral(‘photo_g,PhotocellIO))
  add(sob, analog-in-peripheral(‘photo_b,PhotocellIO))
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    rgbled.set_a(photo_r > 0.4);
    rgbled.set_b(photo_g > 0.4);
    rgbled.set_c(photo_b > 0.4);
    iop.step();
  }
}
```

- lcd peripheral

```
defn arduino-crystal-ball () :
  val sys = System(‘arduino-crystal-ball)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob,bool-in-peripheral(‘tilt,TiltIO))
  add(sob,lcd-peripheral(‘lcd))
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  Timer timer;
  timer.start();
  lcd.clear();
  lcd.locate(0, 0);
  lcd.puts("Hello World!", 12);
  char c = 'A';
  while (true) {
    if (tilt && timer.read() >= 0.05) {
      timer.reset();
      if (c < 'z')
        c++;
      else
        c = 'A';
      lcd.putc(c);
    }
    iop.step();
} }
```

- sensing buzzer peripheral

```
defn arduino-knock-lock () :
  val sys = System(‘arduino-knock-lock)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sob, analog-in-peripheral(‘vibration, SensingBuzzer{_,Res}))
  add(sob, servo-peripheral(‘servo, ServoJITPCB))
  add(sob, bool-in-peripheral(‘button, SWio))
  add(sob, bool-out-peripheral(‘red_led, AnyLEDio{_, LED-DIP, Res}))
  add(sob, bool-out-peripheral(‘green_led, AnyLEDio{_, LED-DIP, Res}))
  add(sob, bool-out-peripheral(‘yellow_led, AnyLEDio{_, LED-DIP, Res}))
  add(sys, sob)
```

```
int main (int argc, char* argv[]) {
  iop.init();
  bool locked = 0;
  ...
  while (true) {
    iop.step();
    if (locked == 0) {
      if (button) {
        servo.set(0.5);
        red_led = 1;
        green_led = 0;
      }
    } else {
      ...
    }
  }
}
```

- buzzer peripheral
- light-sensor peripheral

```
defn arduino-light-theremin () :
  val sys = System(‘arduino-light-theremin)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sob, analog-in-peripheral(‘light_sensor,PhotocellIO))
  add(sob, buzzer-peripheral(‘buzzer))
  add(sys, sob)
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    buzzer.set(.5);
    buzzer.set_frequency(1000 - light_sensor * 900);
    wait(.01);
    iop.step();
  }
}
```

- analog-in peripheral

```
defn arduino-love-o-meter () :
  val sys = System(‘arduino-love-o-meter)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, bool-out-peripheral(‘led_0, LEDio))
  add(sob, bool-out-peripheral(‘led_1, LEDio))
  add(sob, bool-out-peripheral(‘led_2, LEDio))
  add(sob, analog-in-peripheral(‘temperature,TempSensorIO))
  sys
```

```
void set_led(int which, int val) {
  switch (which) {
    case 0: led_0 = val; break;
    case 1: led_1 = val; break;
    case 2: led_2 = val; break;
  }
}
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    float voltage = 3.3f * temperature.value;
    float temp_degrees = (voltage - 0.5f) * 100.0f;
    int num_leds_on = (int)(temp_degrees - baselineTemp) / 2;
    for(int i = 0; i < 3; i++)
      set_led(i, (i < num_leds_on) ? 1 : 0);
    iop.step();
  }
}
```

- servo peripheral

```
defn mood-cue () :
  val sys = System(‘mood-cue)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, servo-peripheral(‘servo, ServoJITPCB))
  add(sob, analog-in-peripheral(‘input, AnyPOTio{_, POT, Cap}))
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    iop.step();
    servo.set(input);
  }
}
```

- dc motor peripheral

```
defn arduino-motorized-pinwheel () :
  val sys = System(‘arduino-motorized-pinwheel)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sob, bool-out-peripheral(‘motor, DCMotor))
  add(sob, bool-in-peripheral(‘button, SWio))
  add(sys, sob)
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    iop.step();
    motor = button;
  }
}
```

- captouchpad peripheral

```
defn arduino-touchy-feely-lamp () :
  val sys = System(‘arduino-touchy-feely-lamp)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sys, sob)
  add(sob, bool-out-peripheral(‘led, LEDio))
  add(sob, captouchpad-peripheral(‘cap_touch_pad))
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  while (true) {
    led = cap_touch_pad.sense() > 10 ? 1 : 0;
    iop.step();
  }
}
```

- hbridge + potentiometer peripheral

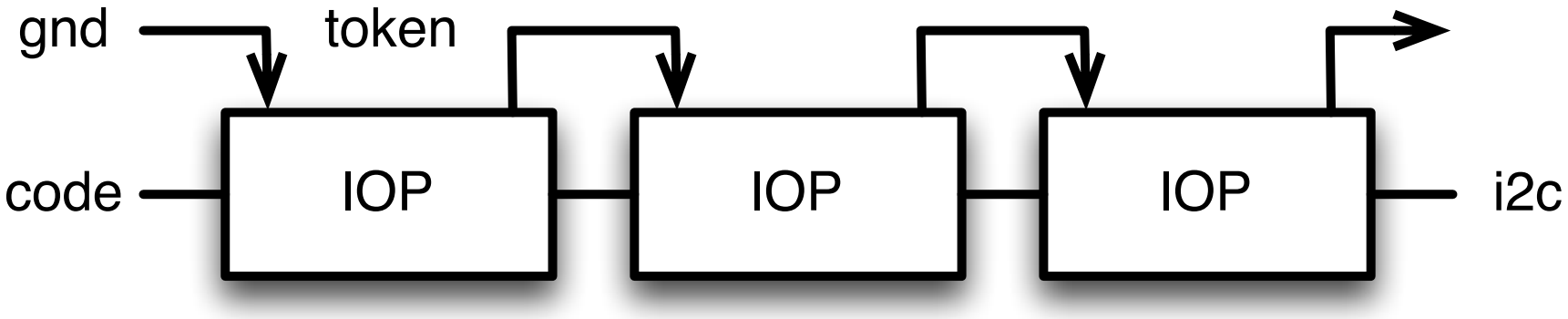
```
defn arduino-zoetrope () :
  val sys = System(‘arduino-zoetrope)
  val sob = SOB(nucleo32-l432kc-dip-rep, true)
  add(sob, l293d-peripheral(‘h_bridge))
  add(sob, analog-in-peripheral(‘speed, AnyPOTio{_, POT, Cap}))
  add(sob, bool-in-peripheral(‘pwr_button, SWio))
  add(sob, bool-in-peripheral(‘dir_button, SWio))
  add(sys, sob)
  sys
```

```
int main (int argc, char* argv[]) {
  iop.init();
  bool dir = 0;
  Timer timer; timer.start();
  while (true) {
    iop.step();
    if (dir_button && timer.read() >= 0.1) { dir = !dir; timer.reset(); }
    if (pwr_button) {
      if (dir == 1) {
        h_bridge.setI1(1); h_bridge.setI2(0);
      } else {
        h_bridge.setI1(0); h_bridge.setI2(1);
      }
      h_bridge.setEN12(speed/4.0);
    } else {
      h_bridge.setEN12(0.0);
} } }
```

- peripherals created to support arduino getting started projects

```
public defn rgbled-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn lcd-peripheral (name:Symbol) ...
public defn led-transistor-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn rotary-encoder-peripheral (name:Symbol) ...
public defn servo-peripheral (name:Symbol, hw:Symbol -> PeripheralModule) ...
public defn buzzer-peripheral (name:Symbol) ...
public defn captouchpad-peripheral (name:Symbol) ...
public defn l293d-peripheral (name:Symbol) ...
```

- programmed over I2C
- code organized by IO processor from low to high with cpu-id.elf
- token passed down i2c bus programming processors in order
- IOP with token gets programmed



Expert JITPCB

How to define:

- systems with precise placement of peripherals
- circuit board generators
- apps that talk to your boards

How to define your own:

- peripherals
- packages
- cpus

How to use the:

- autorouter
- circuit viewer
- bootloader
- assembly instructions
- BOM

Logistics

- Milling lab out
- Get electronics hands on by next Thursday
- OMC here in section monday 4-5p
  - please mill before hand if possible

Next Time

- Circuits Part One

- JITPCB, by Jonathan Bachrach, David Biancolin, Austin Buchan, Duncan Haldane, Richard Lin
- Arduino Projects Book, by Arduino included in Arduino Getting Started Kit