

Sampler

Applications (samplers, drivers) need access to hardware resources like IO ports, digital IO and analog inputs.

pow_on(power_port *pow)

To make life easier for application developers (driver writers), the interface to these would be straightforward, and hide the underlying implementation...

Power Port API

But there may be several different implementations...

...and implementations may use different sets of hardware resources to cover the API

And it ought to do something sensible if the given implementation doesn't support part of the API

SNode Power

pow_on

pow_off

pow_state

pow_read

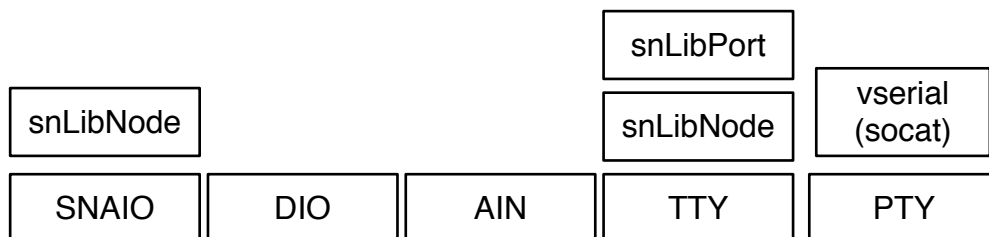
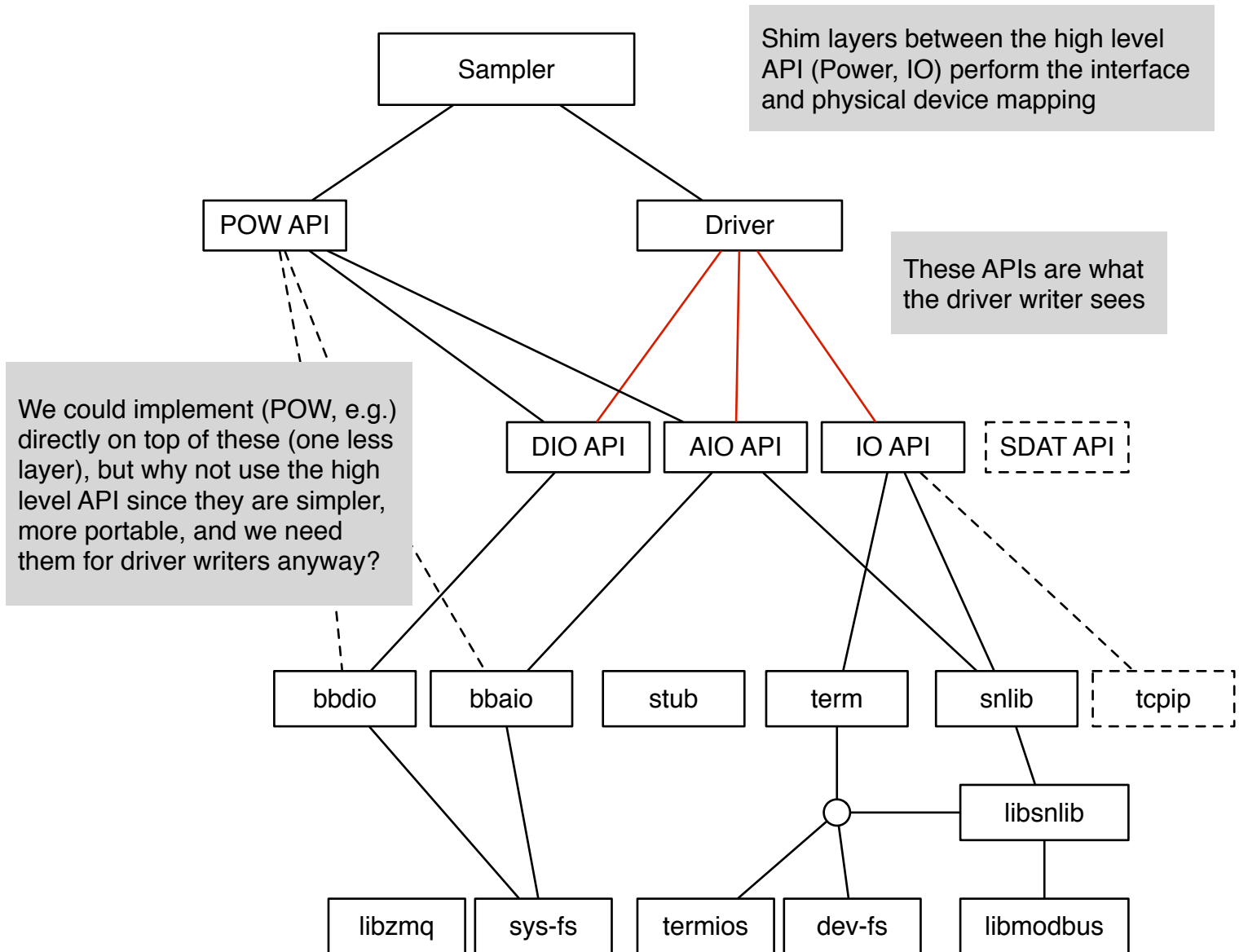
SNPort Power

pow_on
pow_off
pow_state

A mapping is needed to hide the underlying implementation, and to enable an API to use multiple resources.

gpio.0 gpio.1 gpio.2 ain.0

SNode.0
SNPort.0.1



Sampler

An application should use a consistent interface to perform a set of operations, and not worry about the underlying implementation

Power Port

Driver

pow_on()

io_fgets()

dio_set_val()

aio_volts()

POW API

IO API

DIO API

AIO API

logical resources (devices)

uniform API

pow-rser

io-rser

dio-bbdio

aio-bbaio

pow-dio

io-term

dio-snlib

aio-snlib

pow-aio

io-tcpip

dio-stub

aio-stub

pow-stub

io-stub

resource mapping

implementation-specific

bbone-gpio

bbone-aio

termios

libsnlib

libczmq

libmodbus

resources

