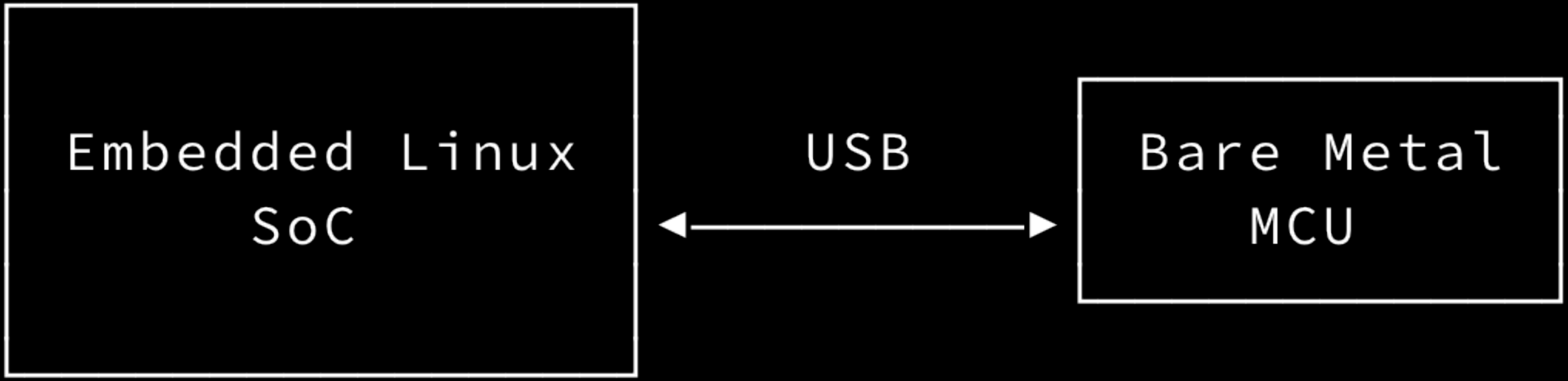
the embedded "buddy system"

or:

james' cheat codes for low/mid volume + rapid embedded development

if you're building something embedded TODAY

consider the
"buddy system"


Embedded Linux
SoC
USB
Bare Metal
MCU

don't do it
all on one chip

embedded linux for the big stuff

easy mode (linux):

networking, file systems, databases, updates, hiring

bare metal for the little stuff

easy mode (mcu):

custom hardware,
real-time, i/o,
low power

use rust for both

easy mode (rust):

share code, tools, workflows, devs

tie it together with postcard-rpc and poststation

easy mode (comms):

usb, uart, spi, i2c

you can have the best of both worlds

if you aren't making 1-10k units (yet)...

the buddy system is probably CHEAPER

development time is

EXPENSIVE

doing mcu things on linux SUCKS

doing linux things on an mcu SUCKS

off the shelf boards
are CHEAP

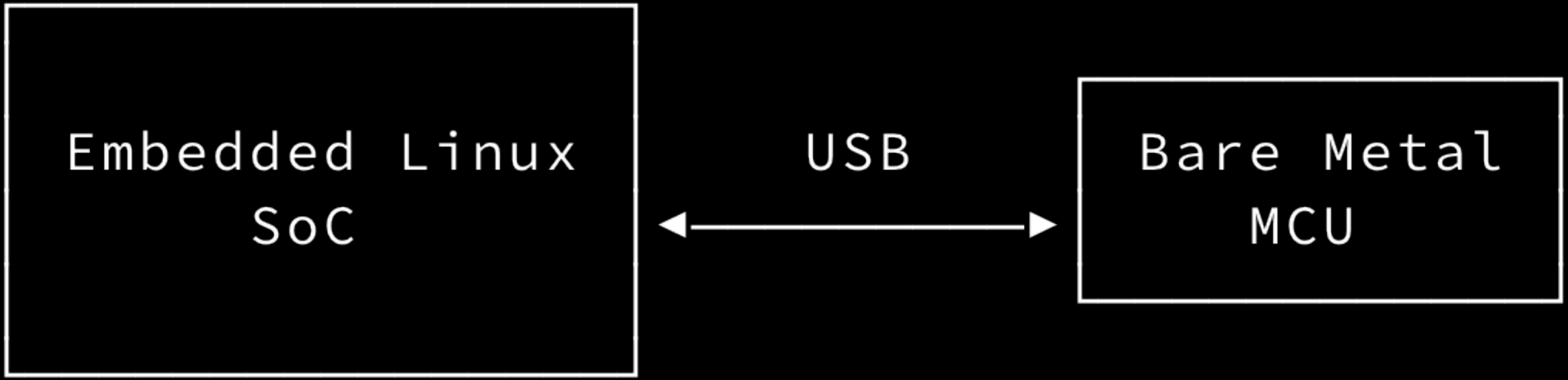
simple boards are
CHEAP

rust:
off the shelf
crates + tools

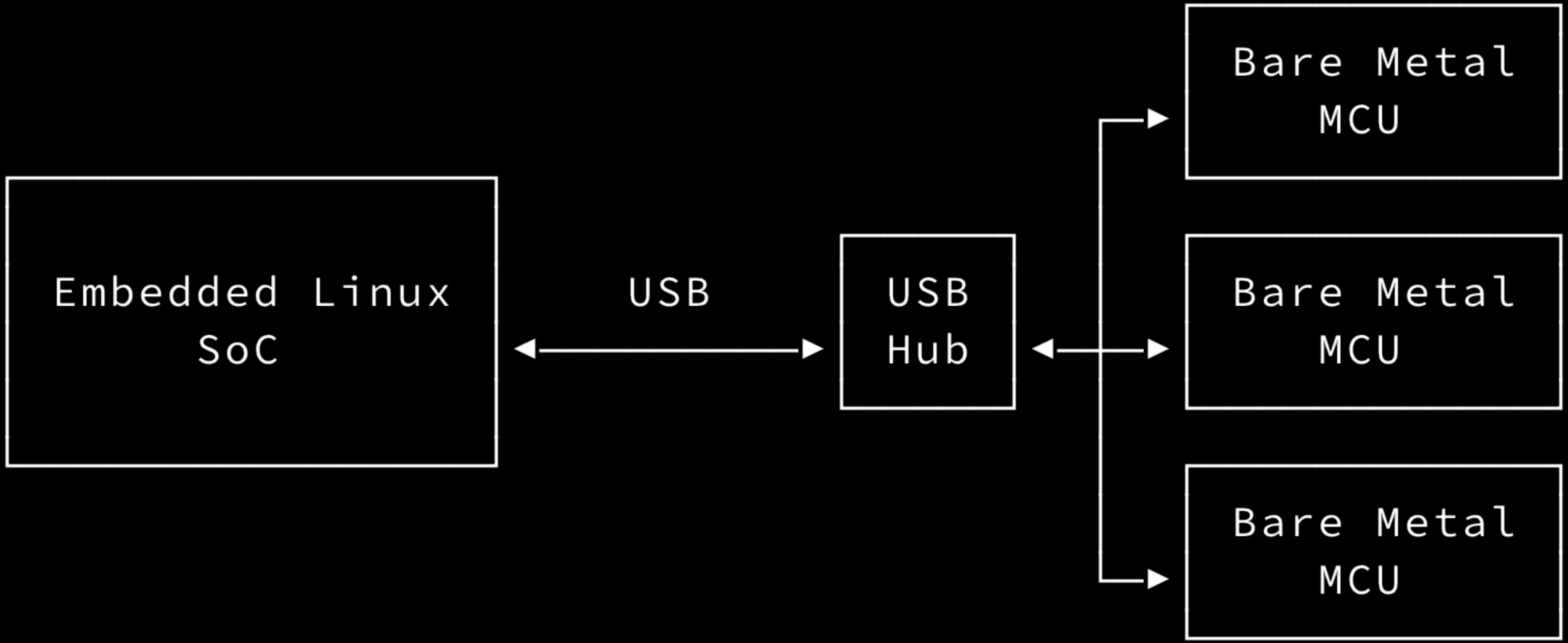
postcard-rpc: off the shelf comms stack

poststation:
off the shelf
tools, apis, sdks

scale horizontally:


Embedded Linux
SoC
USB
Bare Metal
MCU

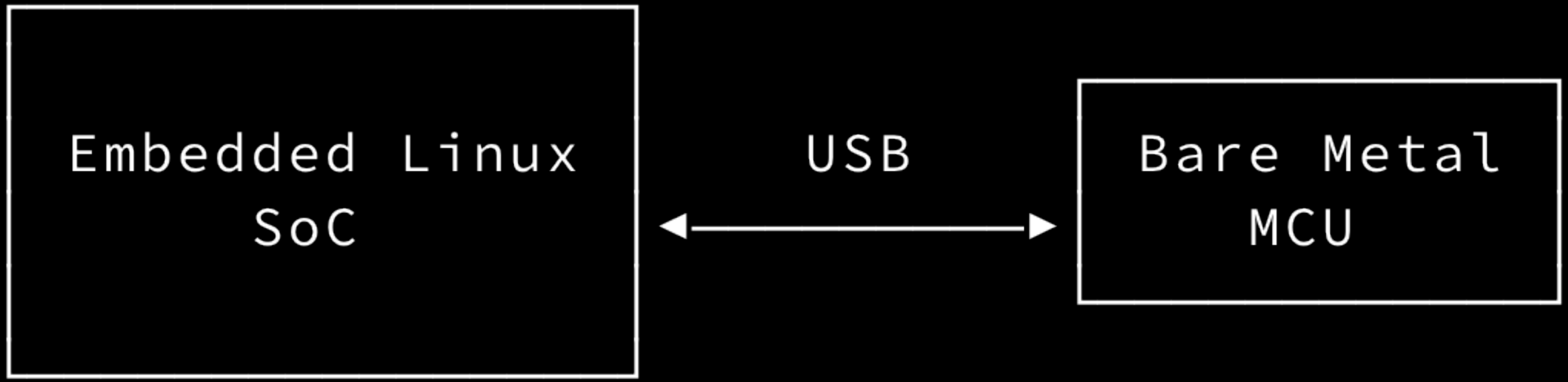
Self-Directed Research

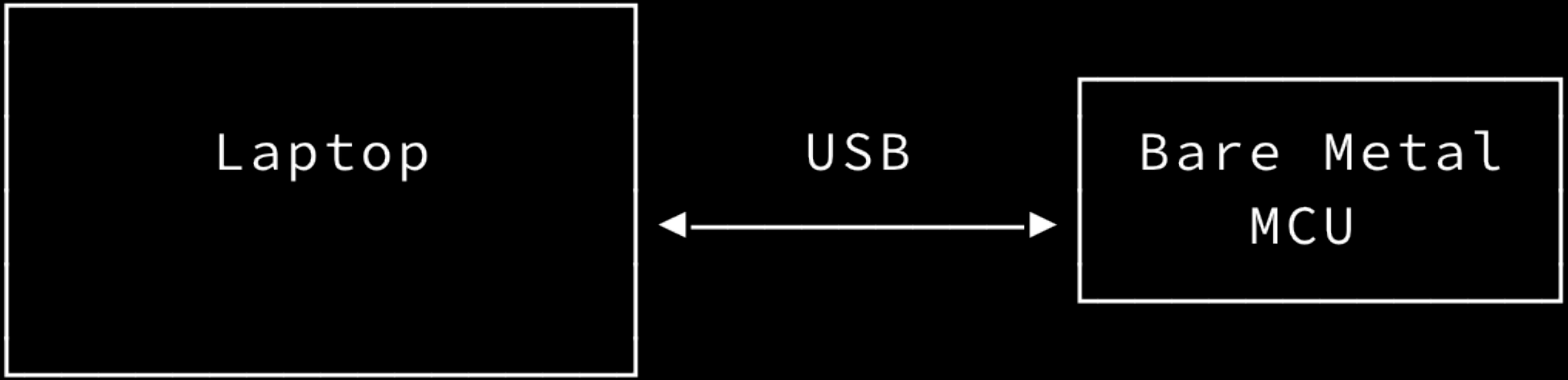


more buddies
more better

iterate faster:

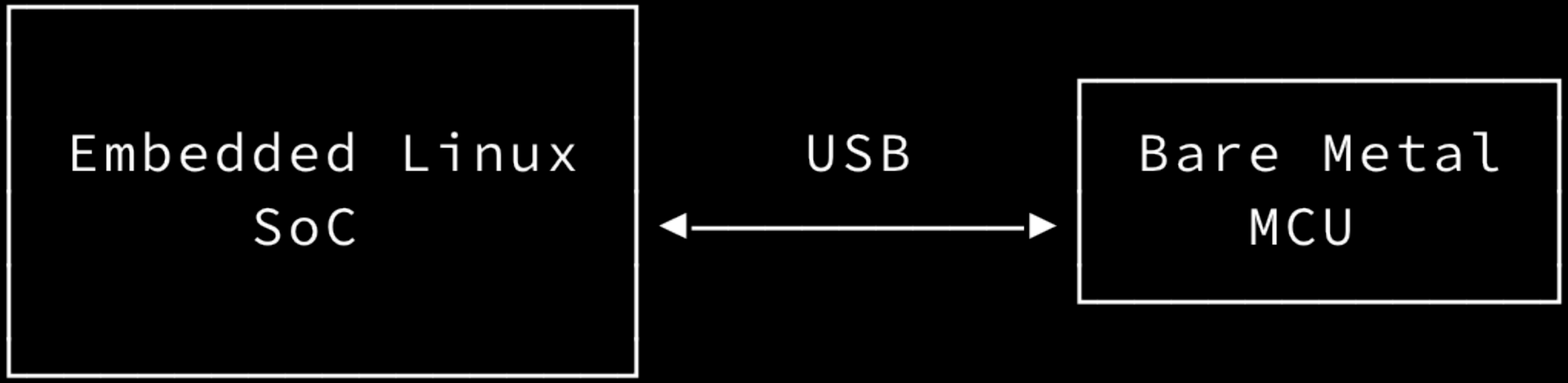
your laptop instead of the SoC

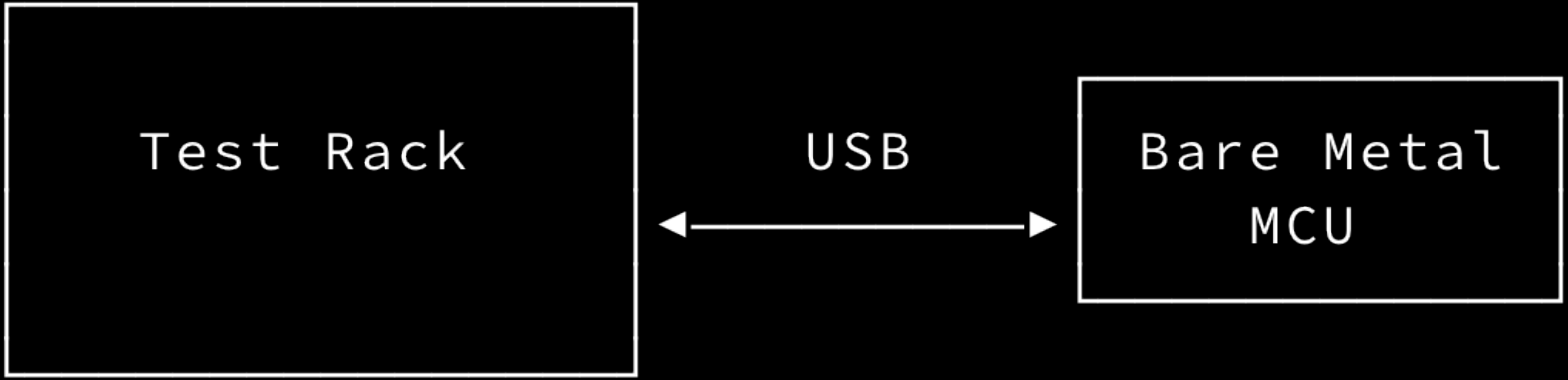

Embedded Linux
SoC
USB
Bare Metal
MCU

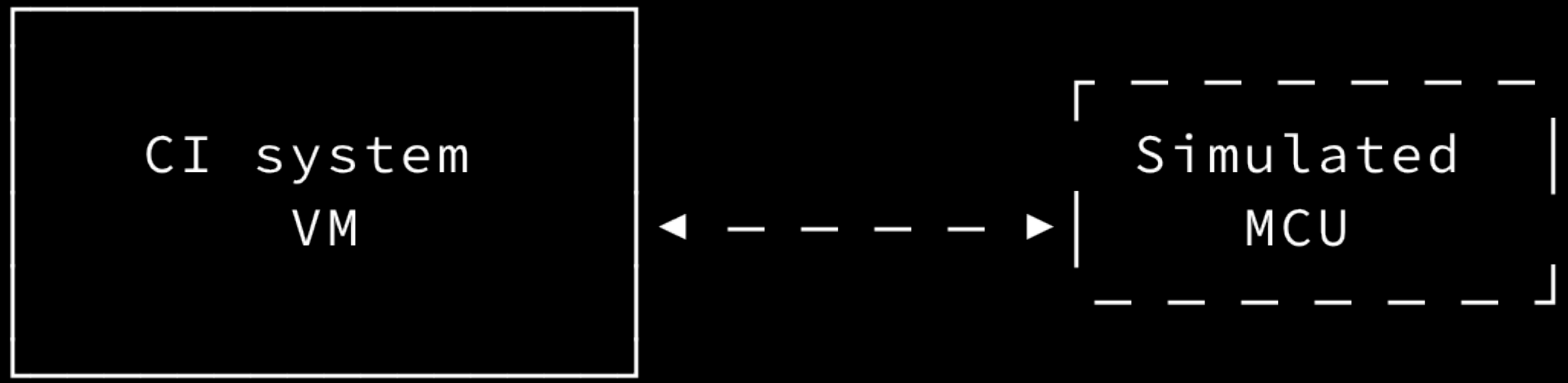

Laptop
USB
Bare Metal MCU

replace either half
as the design evolves

test either half in isolation


Embedded Linux
SoC
USB
Bare Metal
MCU


Test Rack
USB
Bare Metal MCU


CI system
VM
Simulated
MCU

buy yourself time
with things that work
ENOUGH

optimize for cost AFTER shipping v1

you might never NEED v2

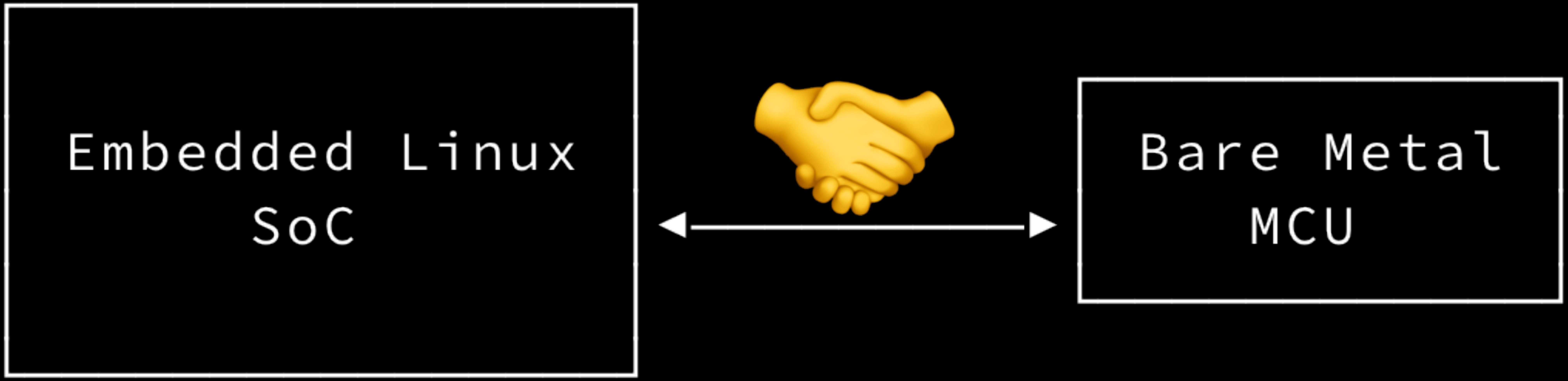
balance the BOM vs the NREs

bill of materials:
per-unit cost

non recurring expenses: design + dev time

treat your buddy like a partner

not like a black box


Embedded Linux
SoC
🤝
Bare Metal
MCU