Embedded Rust Anywhere

Cutting the Cord

How we got here

- We build modern appliances, starting with the June Intelligent oven

- Early software was tied to it's host platform and UI toolkit

- Next generation of appliance software is modular, portable, and scalable

Choosing Rust

- Considered C or C++ as the obvious performant & portable choices.
- Many of our application developers come from Go, Swift, Obj-C and Java.
- We weren't eager for the memory management issues and unpredictable failure modes C or C++ often bring.
- Though young, Rust offered good guarantees and modern syntax.
- After vetting its interoperability story, we were in.

Choosing Rust

- Great cross-compilation support
- Share appliance code with mobile applications
- Configuring a `-sys` crate and build.rs can be easier than working with other build systems.
- Explicit control over memory allocation

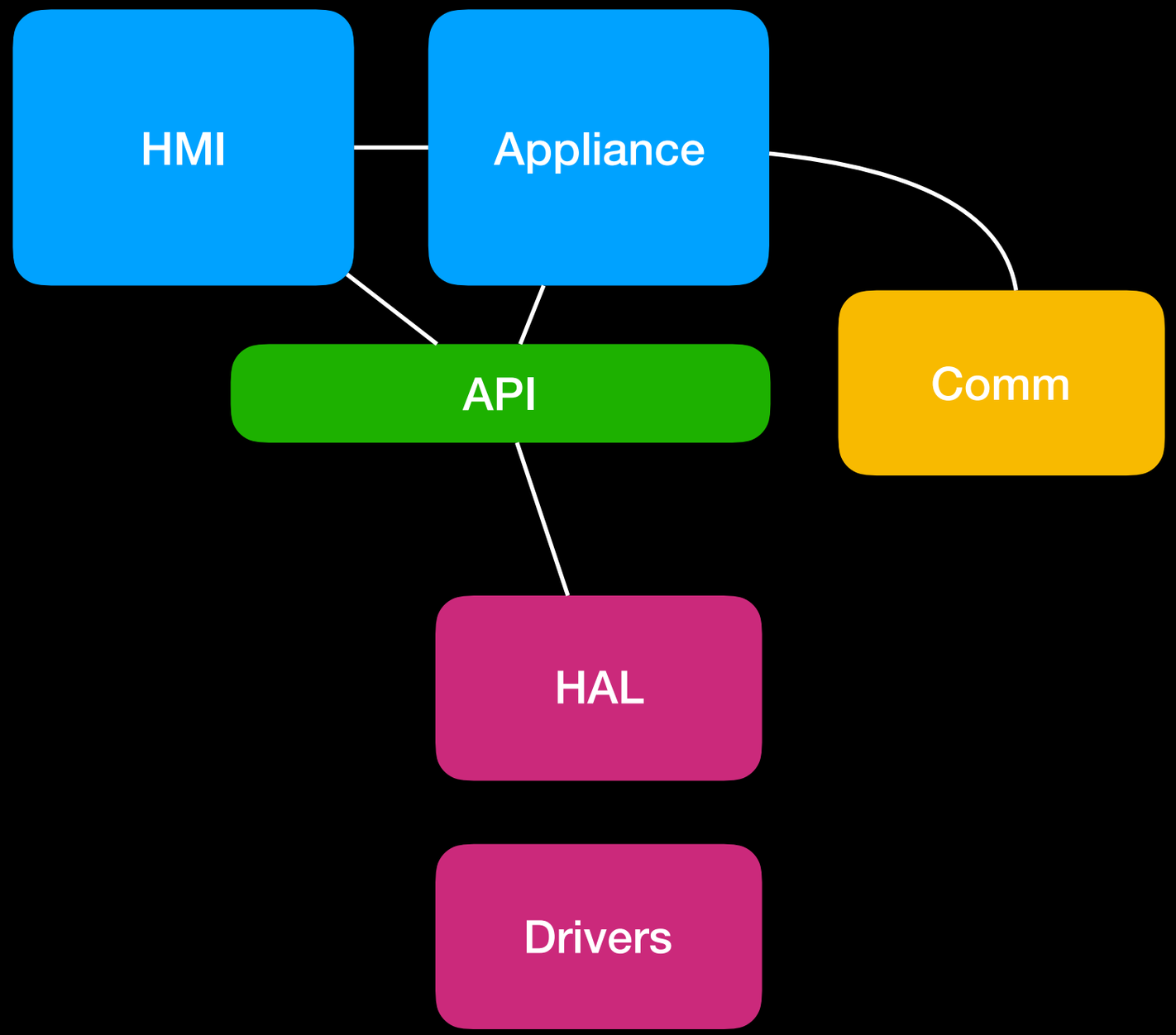
A Whole New System

- Our first new appliance is an embedded system, no_std
- Vendor board support is in C, and they don't support Rust
- Existing embedded team is comfortable doing board bringup in C
- To reduce risk, we brought the system up in C

Divide and conquer

- Divided the system into modules, interfaces defined in C
- Appliance and HMI modules in Rust
- Drivers & HALs in C
- Networking and storage in C
- Module interfaces free of platform-specific types

Divide and conquer



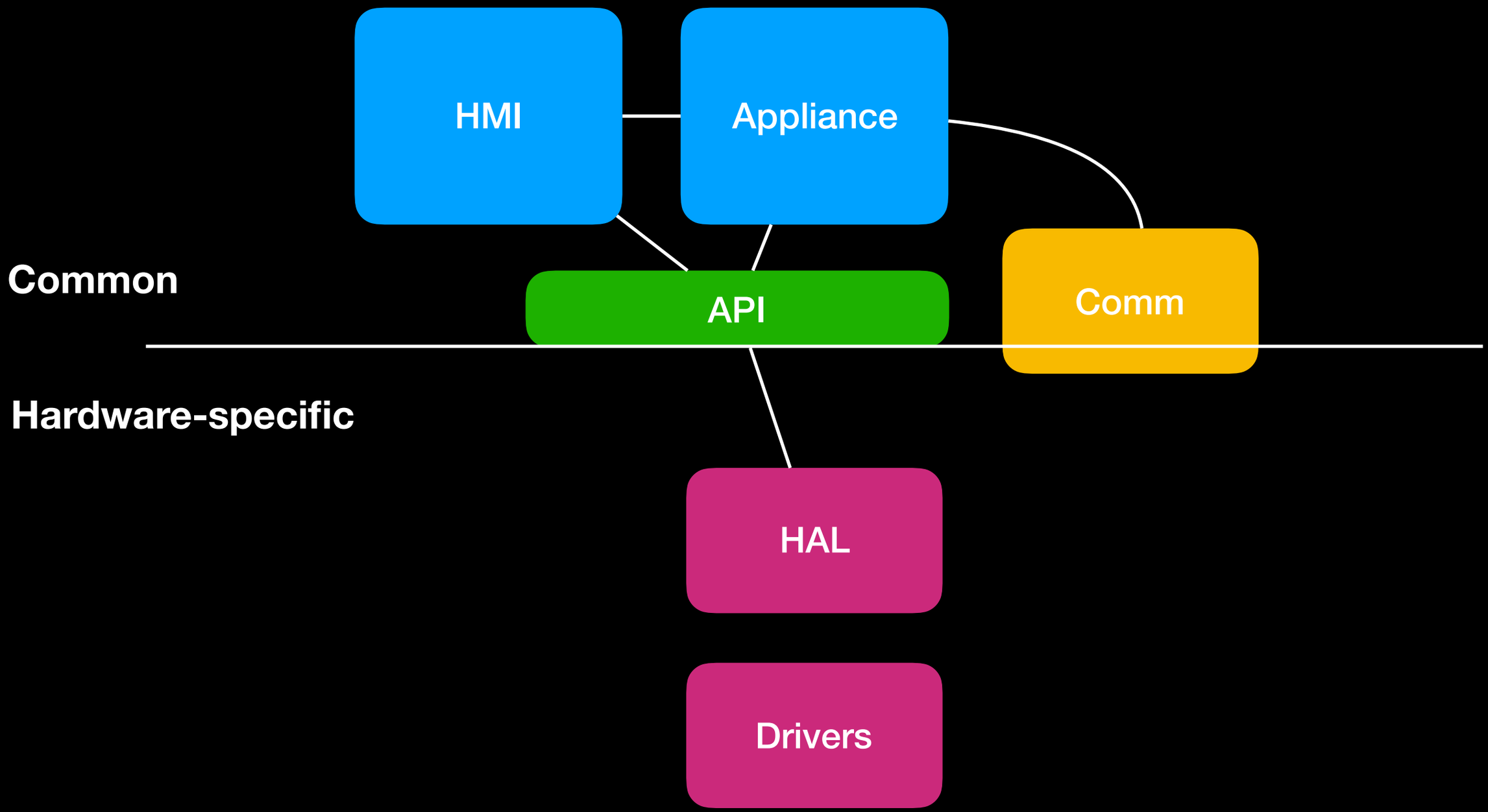
Divide and conquer

```
uint8_t appliance_on_message(uint8_t *ptr, uint16_t length);

bool comm_send_status(uint8_t *ptr, uint16_t length);

void hmi_on_action_button_pressed(void);
void hmi_on_appliance_status(appliance_status_t *);
void hmi_update(void);

void hal_set_led(led_t id, led_state_t state, effect_t effect);
void hal_set_display(char *chars, uint8_t symbols, effect_t effect);
void hal_get_temp_data(thermometer_t *thermometer);

uint32_t hal_get_ticks(void);
```

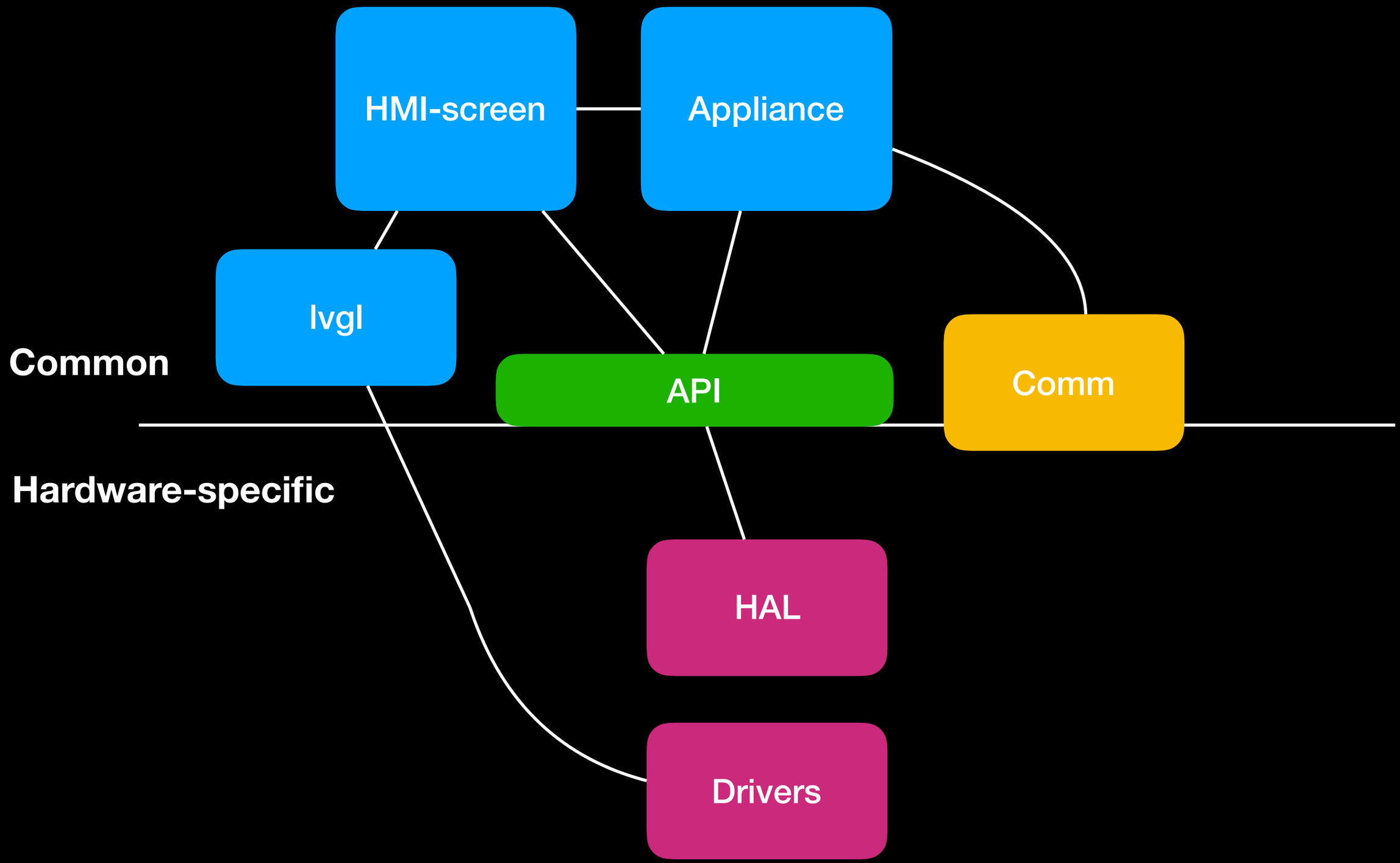
Device and Simulator

- Appliance and HMI logic is purely cross-platform
- HAL implementation rewritten in the simulator - UI widgets in place of hardware
- Device-specific networking and storage code is isolated within a submodule, reimplemented in the simulator

Device and Simulator



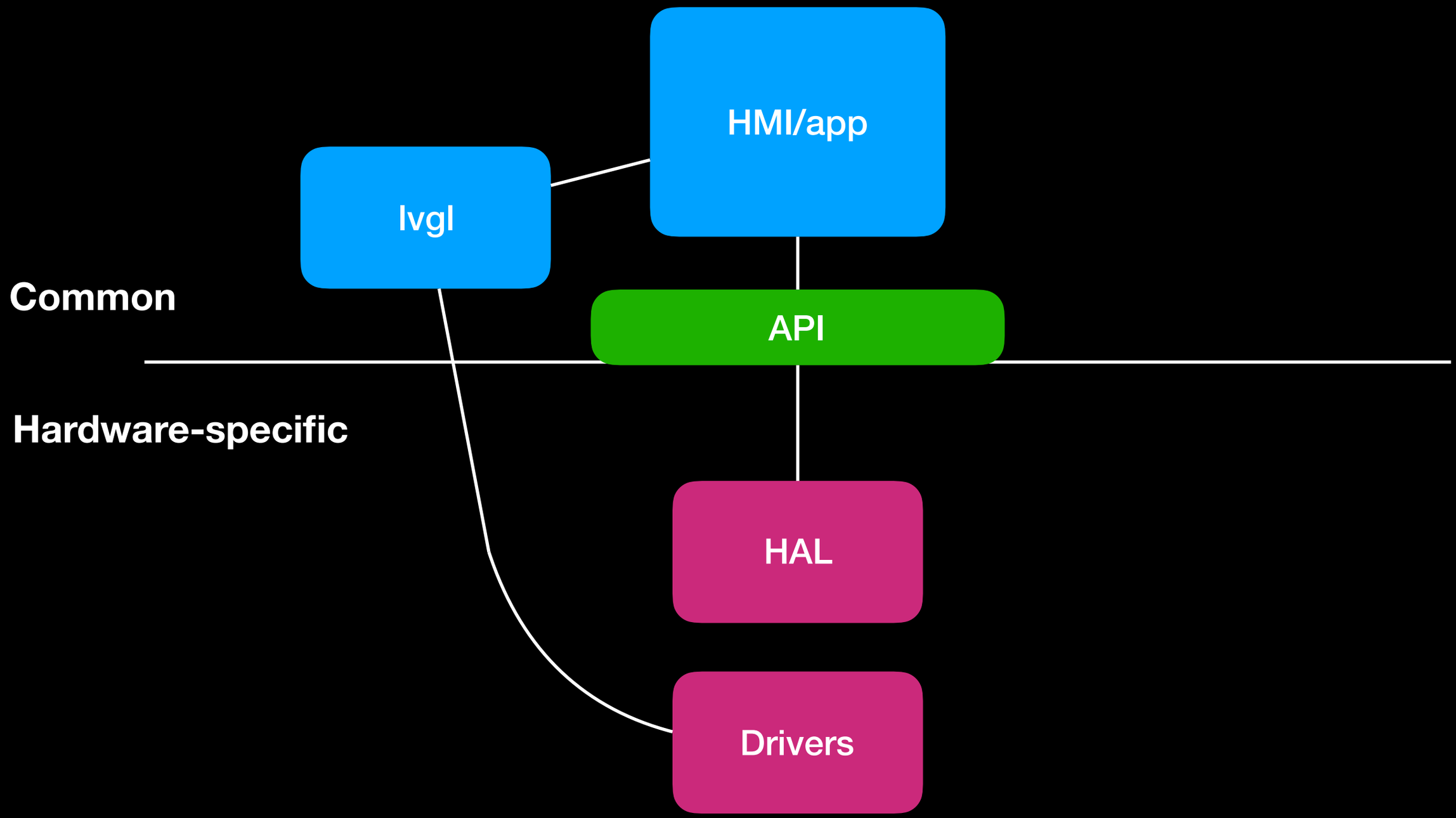
Device and Simulator



Simulator benefits

- Faster than bringup
- Limited hardware
- Simulate hardware I/O
- Simulate network traffic
- Early integration with mobile apps
- Verify tricky hardware edge cases

Sample



Sample

```
// Driver module API
extern void drivers_init(void);
extern void drivers_set_led(LEDs which, bool on);
extern bool drivers_is_button_pressed();

// HMI module API
extern void hmi_init(void);
extern void hmi_update(void);

// LittlevGL bindings
lv_disp_drv_register(&displayDriver)
```

Try it!

- Example repo: github.com/junelife/rust-anywhere
- Targets STM32F429 Discovery board
- Defines API for display, LEDs, and buttons
- Build and run as simulator or on device

Try it!


Rust Anywhere
Stopwatch
0
10
20
30
40
50
60
00:00.00


www.st.com/stm32f4-discovery
STM32F429I-DISC1
MB1075C
Stopwatch
0
10
20
30
40
50
60
00:00.00

A Tour of the Sample

- device
  - Embedded app based on LittlevGL demo app, written in C
  - Provides board bring up and configuration, main application loop
- simulator
  - macOS application, written in Swift
  - Provides simulated UX and hosts shared logic
- crates
  - Rust workspace
  - Contains both shared and platform specific code

Rust Workspace

- Top level static libraries
  - libdevice, libsimulator
- Common application logic
  - api, ffi, lvgl/lvgl-sys, hmi
- Platform specific code
  - board, drivers

Debugging the Simulator

- Rust works great with LLDB and GDB debuggers
- Integration with popular IDEs like Visual Studio Code or CLion
- Xcode plugin support:
  - https://github.com/mtak-/rust-xcode-plugin

Conclusion

- Invaluable to distribute demo software to developers, designers, testers, partners
- Isolates device-specific bugs
- Strong module boundaries make it easier to replace implementations (C with Rust)
- Appliance crates will be reused on devices with a full OS

Thanks!

- Come visit us at the first Impl day!

- We're hiring! juneoven.com/careers