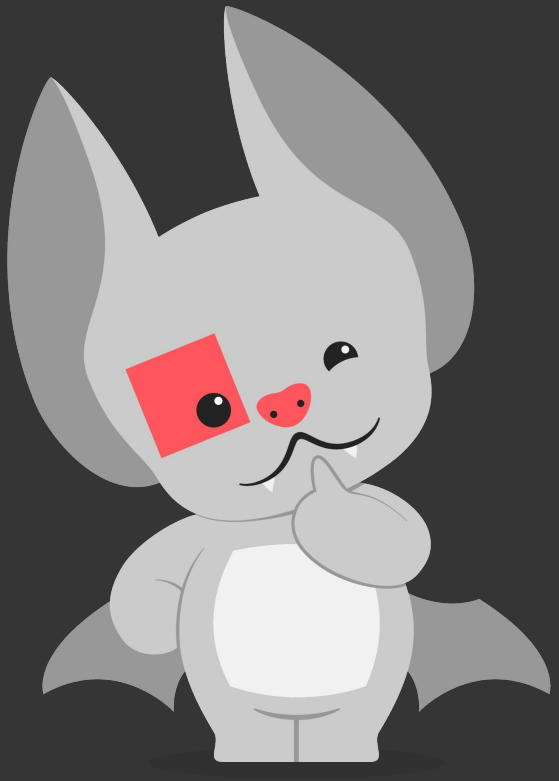
Microservices for Microcontrollers

Composable Software Architecture for Embedded Systems

Um, Aren’t Microservices a Cloud Thing?



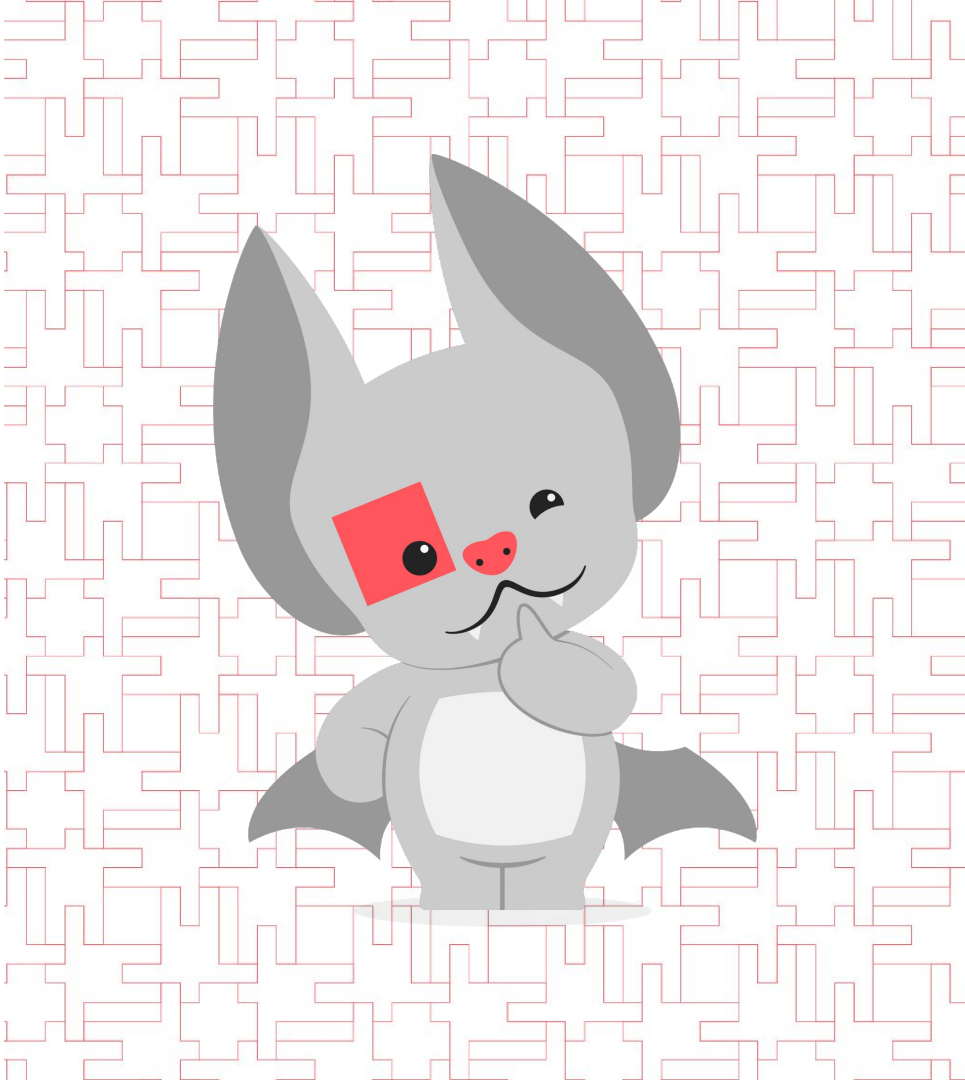
What is a Microservice?

Modular Software

- Fine-grained, loosely coupled, independently deployable software components

Built for Speed

- Organized around business functions. Many teams in a large organization can push updates faster and without fear of breaking everything.



Okay but Firmware?

Modular Software

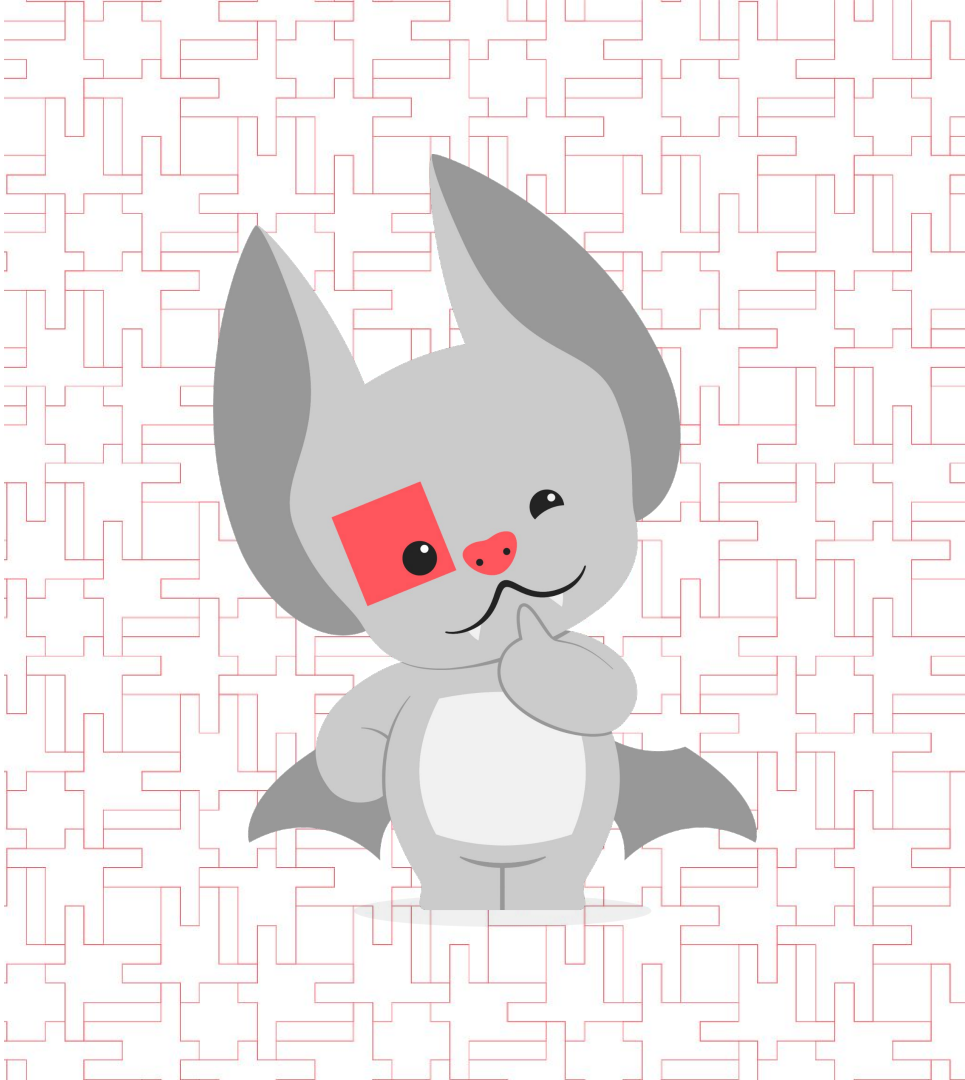
- Independent, loosely coupled, fine-grained tasks communicating through well-defined IPC mechanisms

Improved Encapsulation

- Better code organization makes firmware easier to read, maintain, and test.

Composability

- Smaller pieces are more reusable



What is Composability?

A Lego Set of Components

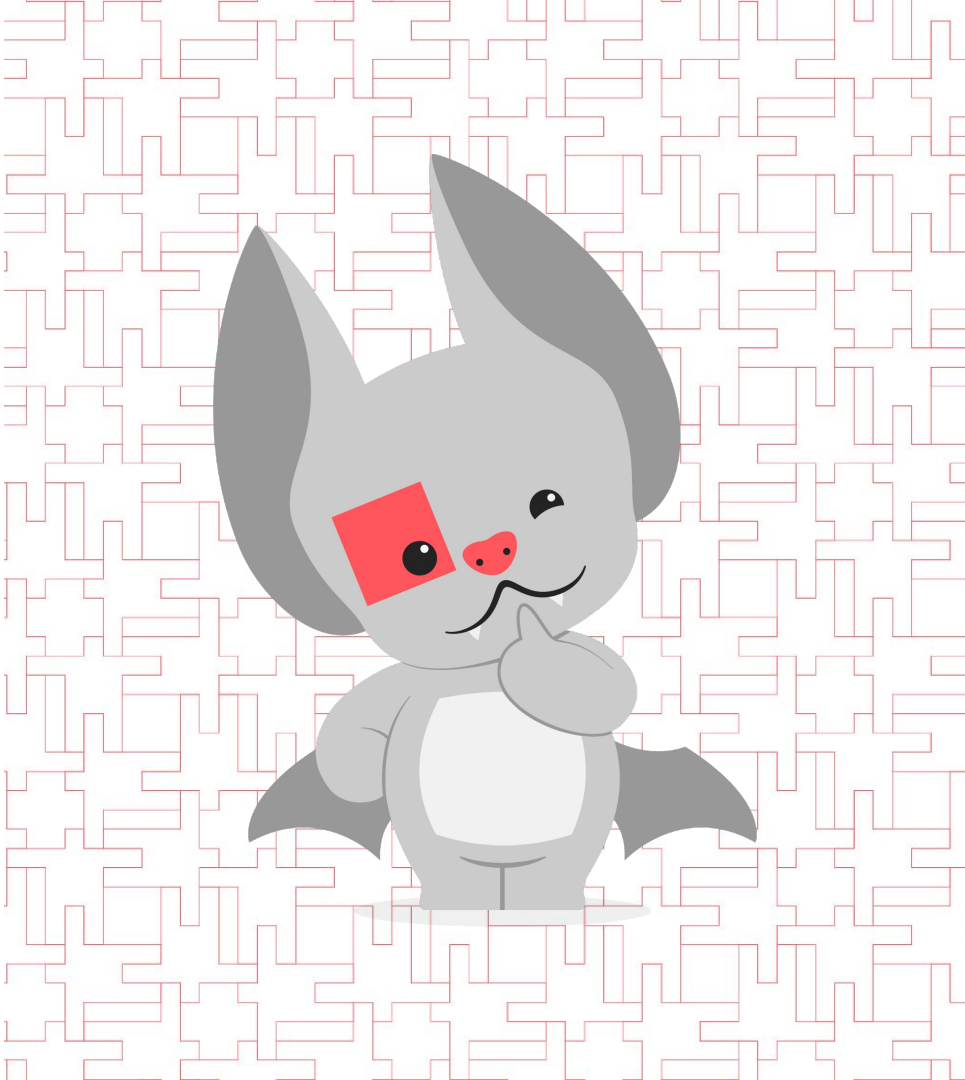
- Small building blocks of functionality that are combined into a larger application

Infinite combinations

- Components can be added, removed, or swapped out to change the functionality of the application, without requiring source code changes

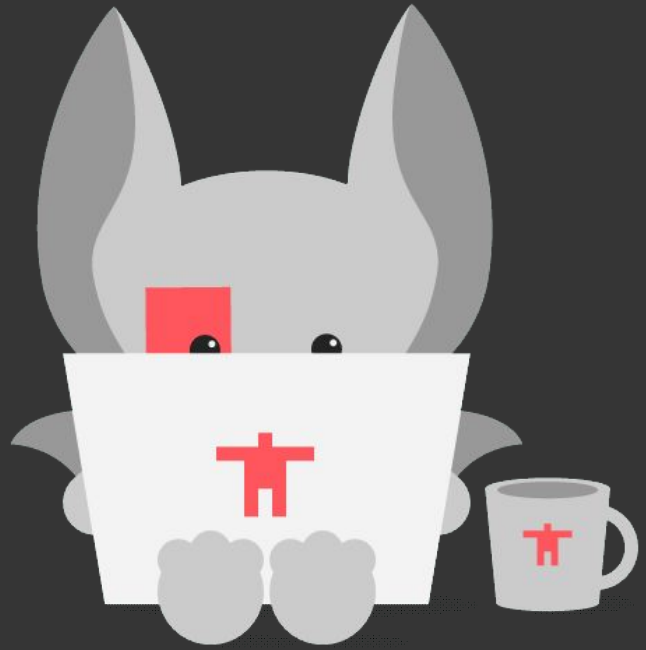
Support Them All

- Create bespoke firmware for each revision of a product by selecting the required components



In Practice

- Tasks
- IPC
- Event Tasks



Tasks

Fine Grained

- Many small tasks
- Each concerned with one (1) business function

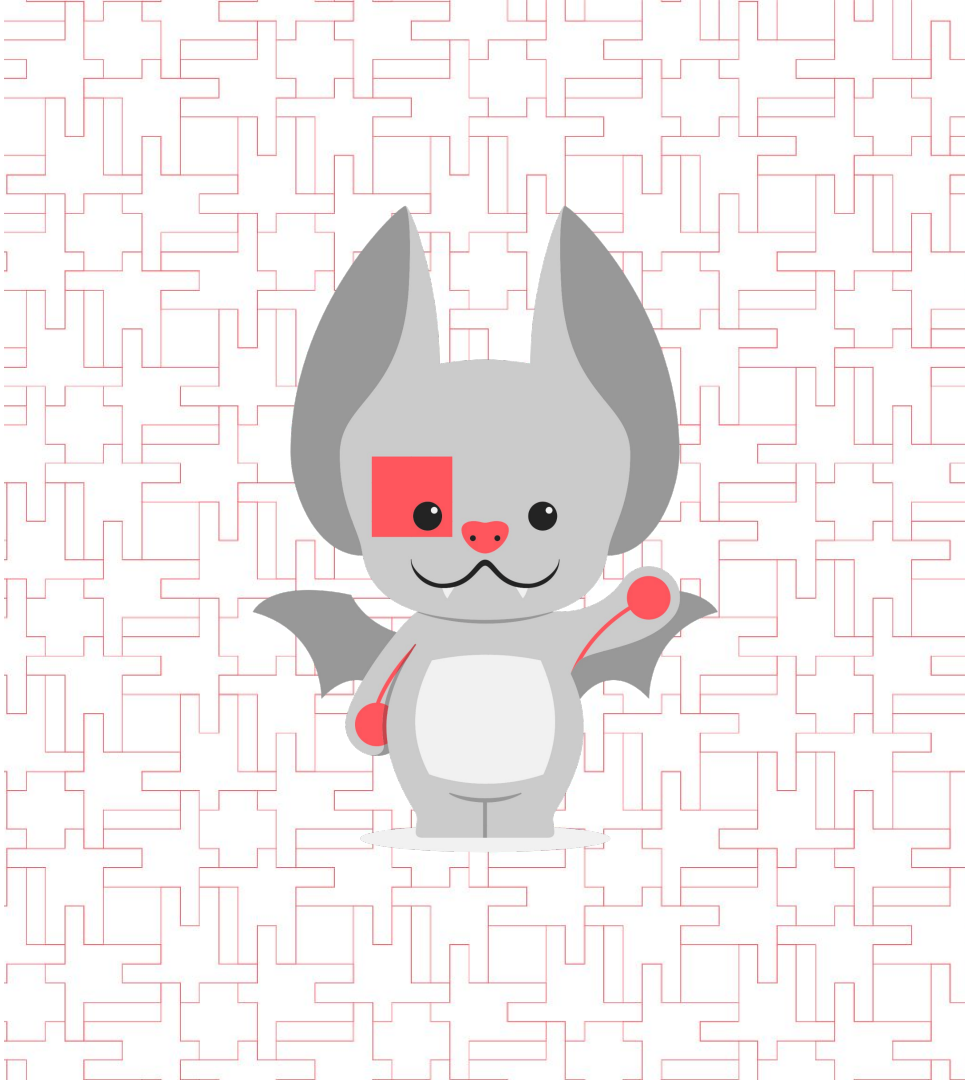
Loosely Coupled

- No public interface
- All communication through indirect IPC

A Collection of Equals

- No main()
- No library/driver initialization

All Threads are Tasks, but Not All Tasks are Threads (more later)



IPC: PubSub

Data-Oriented Architecture

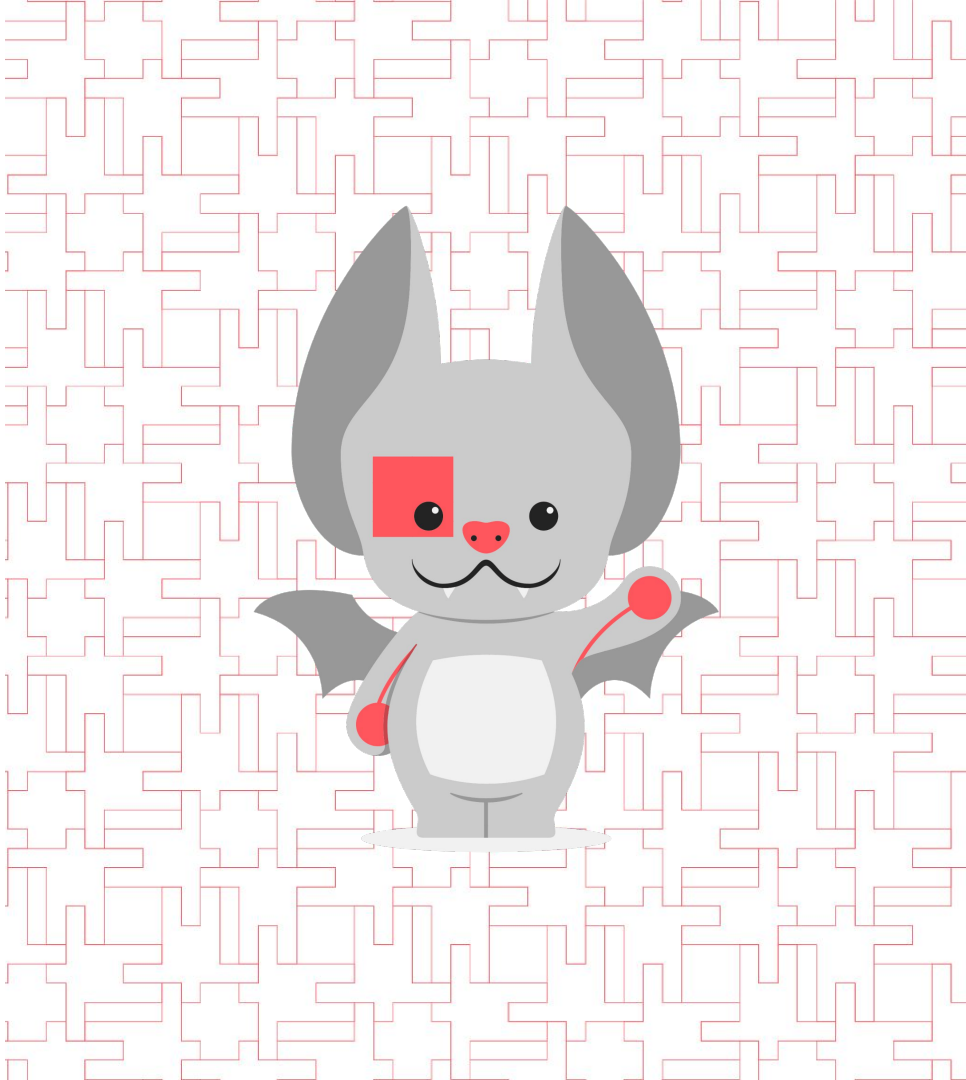
- Focus is on the form and flow of data, not on connections between modules
- Modules communicate without knowing about the other party

Removes inter-task dependencies

- Tasks communicate with Pubsub, not (directly) with each other

Removes development dependencies too

- Publishers and Subscribers easily mocked and tested



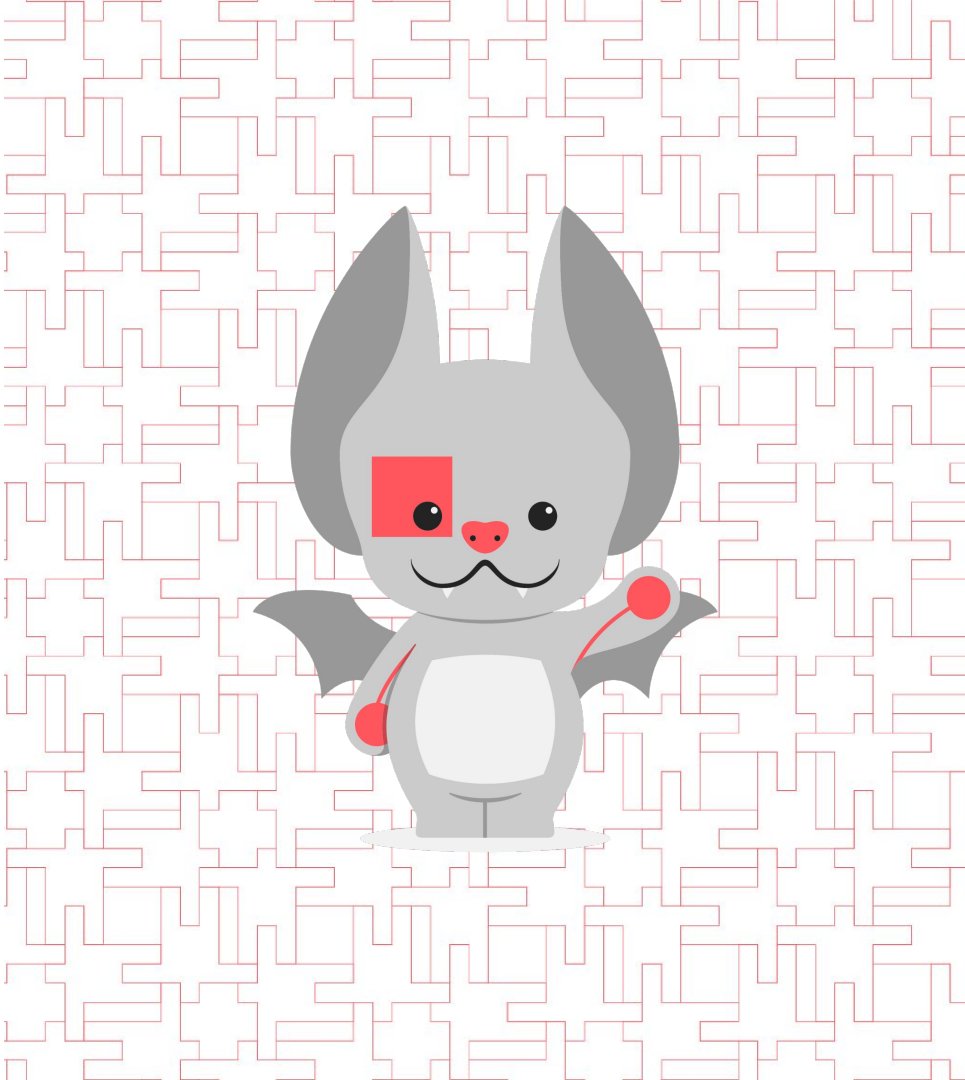
Tasks Redux: Events

Threadless Functions™

- Multiple tasks executing on one thread, running to completion in response to events

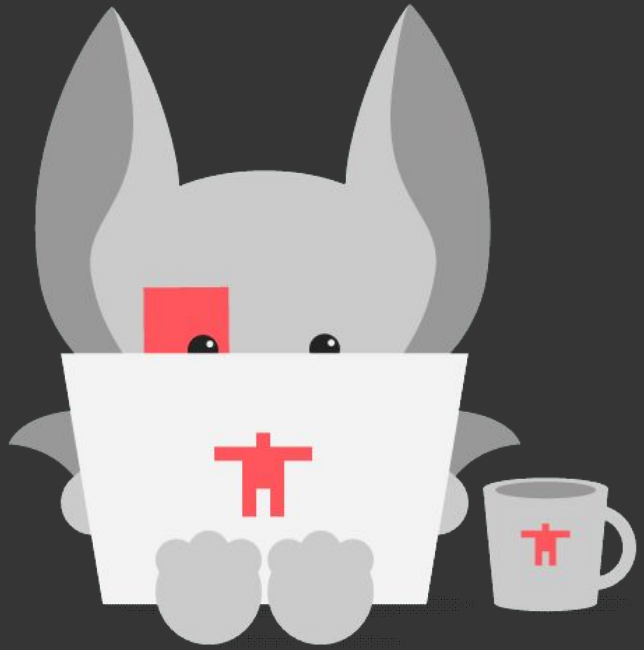
Tread Lightly

- Need to ensure that the event thread isn’t blocked



Building it in Zephyr

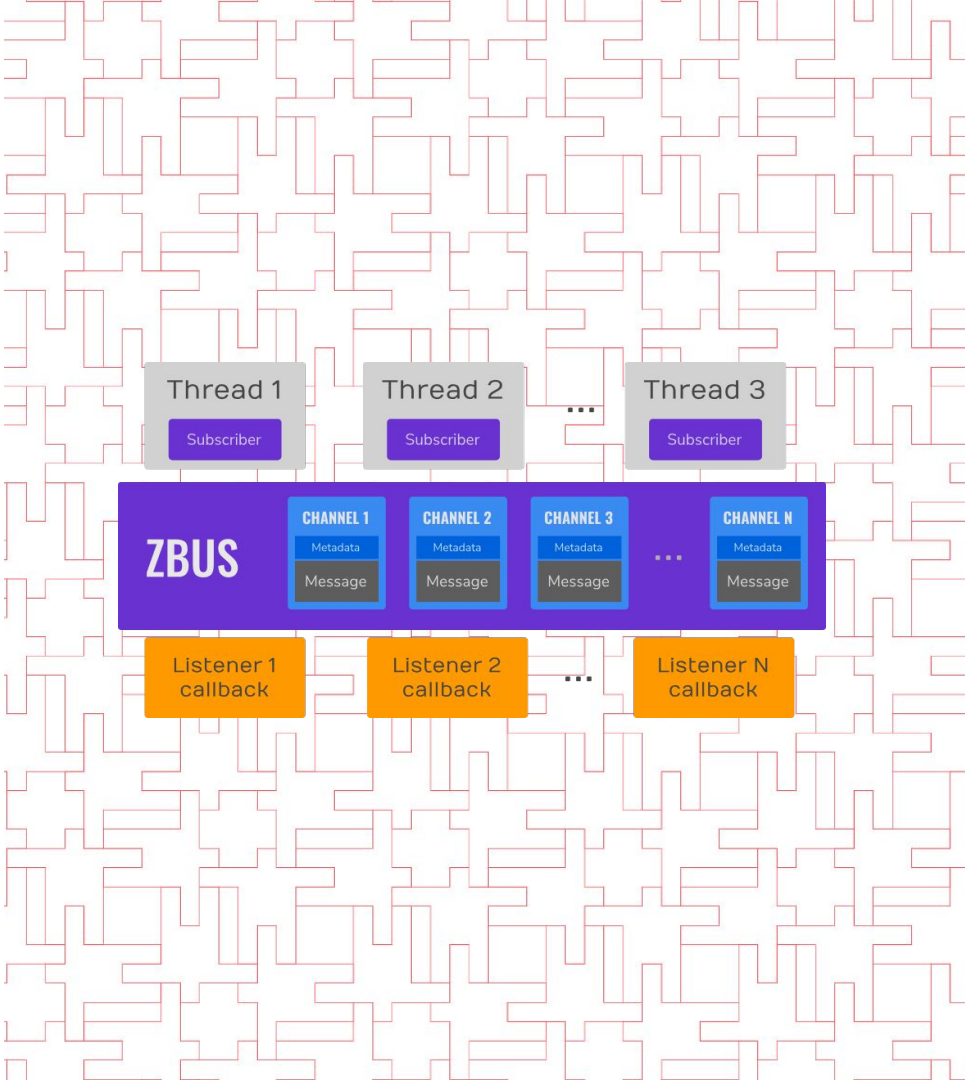
- zbus
- Iterable Sections
- System Initialization



zbus

- Zephyr's native pubsub bus
- Also implements event tasks (listeners)

See the Zephyr Tech Talk on zbus!



Iterable Sections

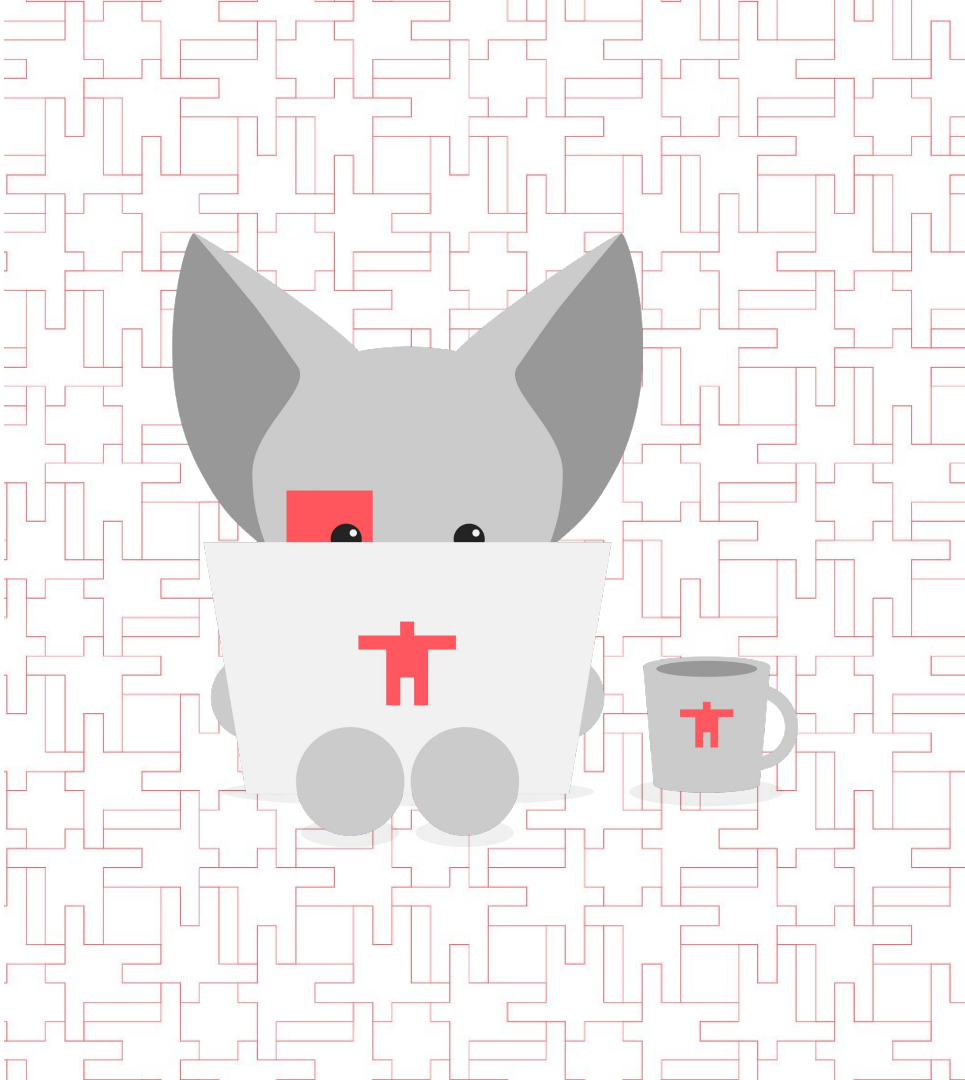
Leverage the Linker

- Allow modules to register themselves. No API call at bootup, no central registry.

Build Time Configuration

- Compiling a file integrates that module into the system.

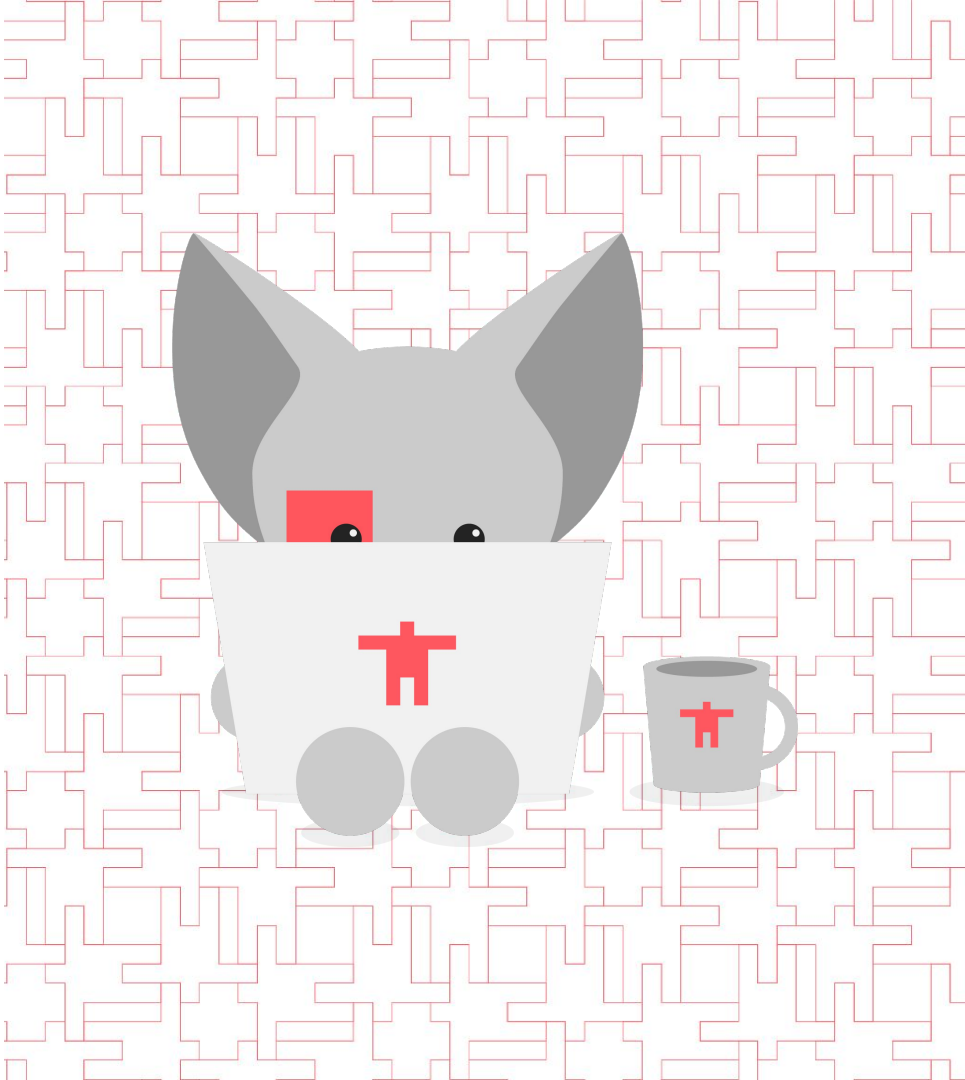
* Remember: no main(), no public APIs!



System Initialization

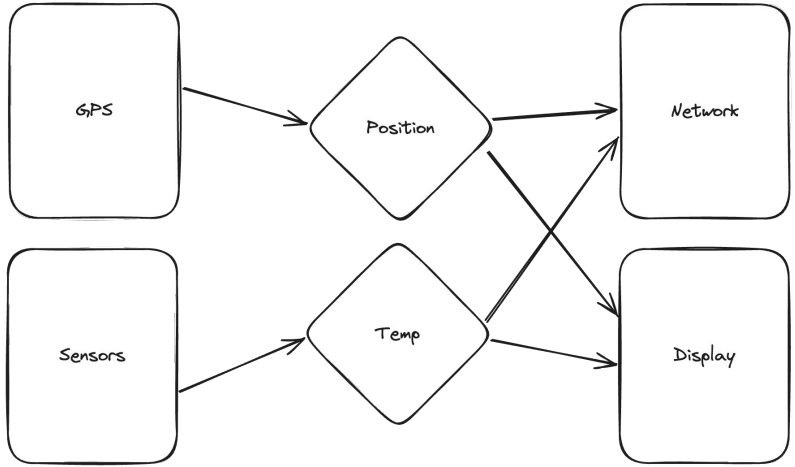
Tasks come and go but initialization is forever

- Zephyr’s system initialization system allows us to decouple initialization of libraries, subsystems and drivers from which tasks are included.



Real World Example

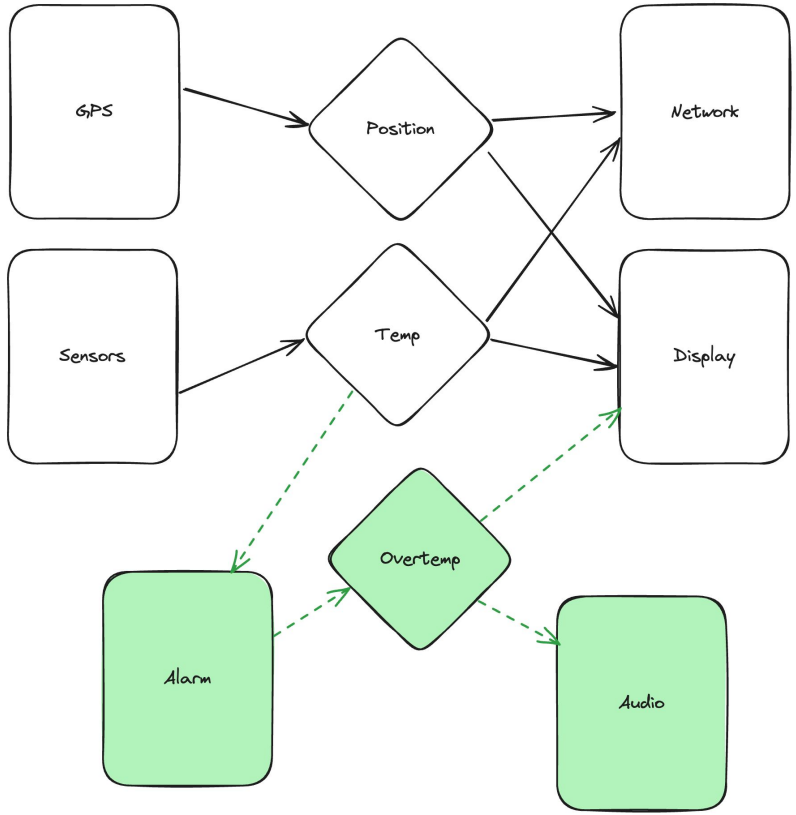
Cold Chain Asset Tracker



New Feature! (temperature alarm)

Customer requests local alerting

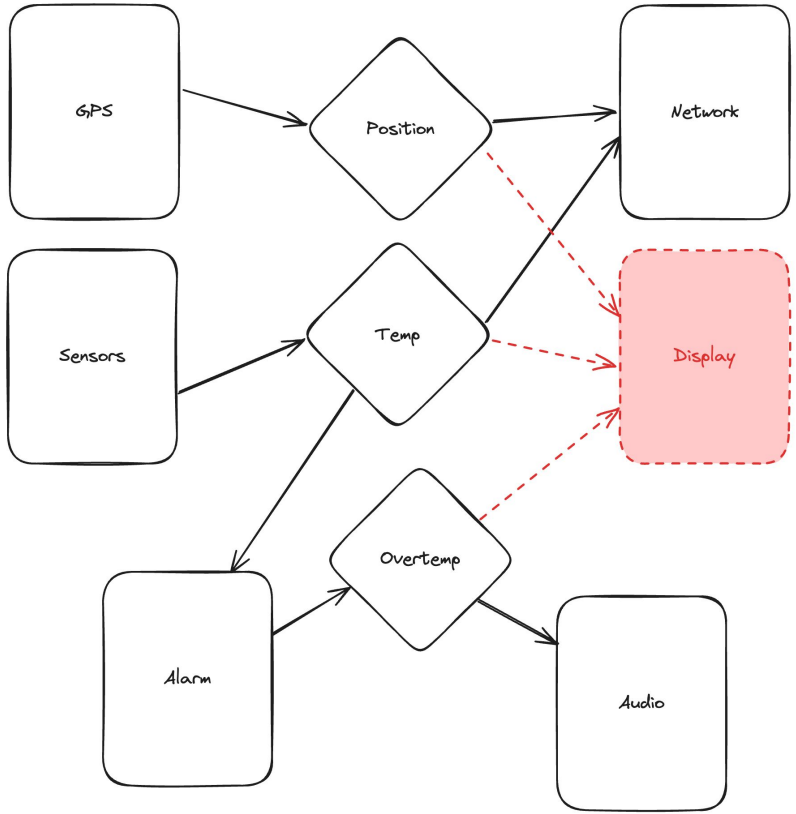
- Alarm Event Task listens to temperature topic, publishes to overtemp topic
- New Audio Thread Task plays tones
- Display Task adds an alert screen



Cost Down (Remove Display)

Display is too expensive, let's remove it!

- Don't compile the Display task



Questions?

golioth