Haskell for Resource-Limited IoT Systems

Comparison of various existing approaches

Aditya Shidhaye

Frankfurt University of Applied Sciences
Supervisor: Prof. Dr. Oliver Hahm

RIOT Summit, 2026

The Haskell fascination

- Not a Haskell wizard
- Not an IoT wizard
- Exploratory incursion
- As of now primarily rooted in literature

Lets go!

The bugs that can pop up in C programs can be hard to identify

- MCU firmware often combines sensors, actuators, timers, radio stacks and low-level C.
- C gives exact hardware control, but many correctness rules remain the programmer's responsibility.
- Haskell can encode more domain distinctions directly in the type system.

C

```
typedef float Meters; typedef float
Seconds;
Meters distance = 10.0f; Seconds time =
2.0f;
distance = time; // legal C
```

Haskell

```
newtype Meters = Meters Float
newtype Seconds = Seconds Float
distance :: Meters distance = Meters 10

time :: Seconds time = Seconds 2
distance = time - type error
```

Three deployment paths

1. Generate C

- Haskell on host
- generated C on device
- no Haskell runtime

Example: Copilot

2. Small runtime

- Haskell subset compiled
- evaluator on device
- heap and GC present

Example: MicroHs

3. Generate hardware

- Haskell-like hardware design
- RTL output
- FPGA/ASIC flow

Example: Clash

They put different artifacts on the target.

Copilot describes streams in Haskell and emits C99.

- Nasa!
- Good fit for: runtime monitors, periodic checks, threshold logic, small control loops.
- Memory usage guarantee
- No Haskell on device, generated C - use it how you want
- Small-device evidence: `arduino-copilot` targets Arduino Uno R3, with 2 KiB SRAM and 32 KiB Flash.
- RTOS evidence: `zephyr-copilot` already provides a Zephyr-oriented route.

MicroHs puts more Haskell on the MCU

- MicroHs compiles a Haskell subset to combinators.
- A compact C evaluator executes those combinators on the target.
- Dynamic allocation, a configured heap and garbage collection are part of the design.
- Current MCU evidence is on larger boards such as STM32F407-class systems and RP2040.
- RTOS integration possible ? (GC pauses ?)

Various possible work

- Flash size of complete firmware images, not only generated source size.
- Control-loop timing: average, worst observed and deadline misses.
- MicroHs GC pause time on real MCU firmware.
- Amount of C glue needed for RIOT, Zephyr or FreeRTOS integration.

Applying Haskell can take various forms

- For the smallest devices, Haskell-hosted C generation is the most credible path.
- For more ordinary Haskell on an MCU, MicroHs is the important current direction, but it brings heap and GC questions.
- For FPGA-style designs, Clash belongs to the hardware layer rather than the MCU firmware layer.

End

Thanks for your time!