WEBINAR
EMBARCADOS

Inscreva-se!

ESCREVENDO UM MICROKERNEL PARA DOMINAR ISOLAMENTO DE MEMÓRIA

Felipe Neves
Engenheiro de Sistemas Embarcados

29 JUL 2026
HORÁRIO: 19h30

MOUSER ELECTRONICS

EMBARCADOS

Patrocinado por

MOUSER ELECTRONICS

Agenda

1. O problema da isolação de memória em sistemas embarcados e a filosofia de um microkernel.
2. O dia em que escrevi meu microkernel para entender melhor o que significa minimizar o kernel space e isolar memoria.
3. Show me the code.

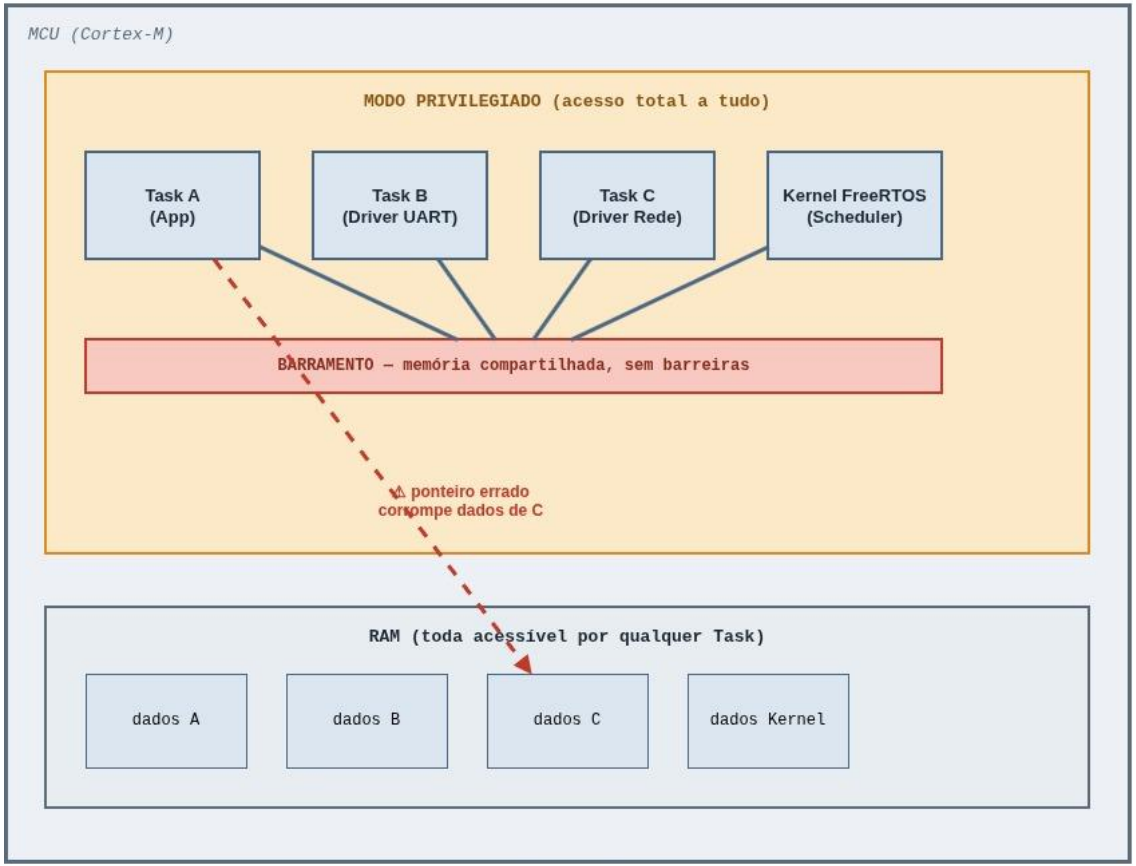
MOUSER ELECTRONICS
EMBARCADOS

Quem sou eu:

- Felipe Neves, Engenheiro de Sistemas Embarcados
- Atualmente pesquisador de doutorado em engenharia mecânica, foco em controle de motores.
- Experiência em setores como Automotiva, Robótica e Semicondutores
- Interesses em topicos como: Controle, Robotica e Computação Heterogenea.

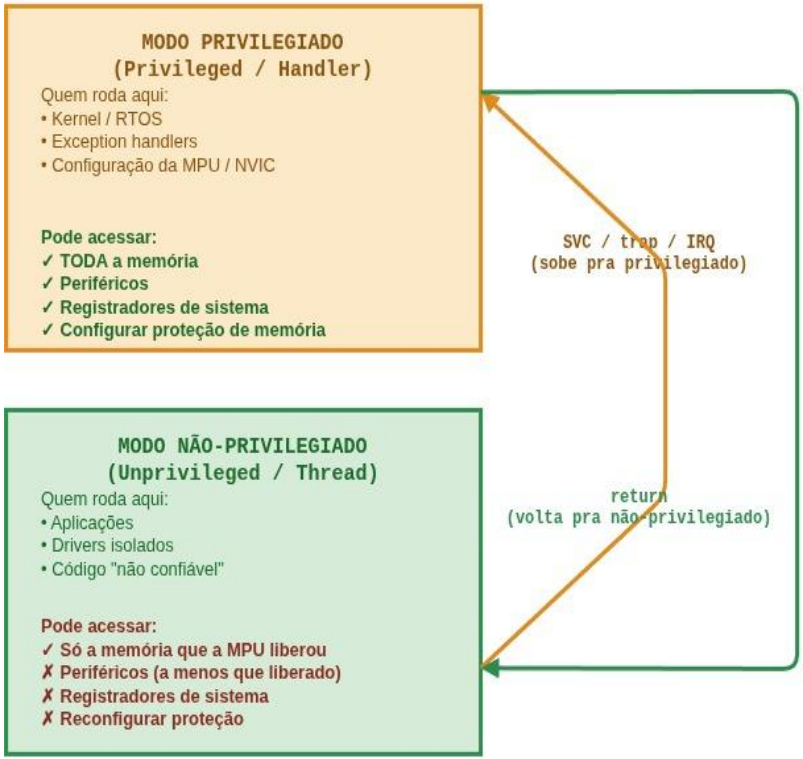
Parte I : O problema de isolação de memória e o microkernel

Firmware Hoje: FreeRTOS — Tudo em Modo Privilegiado



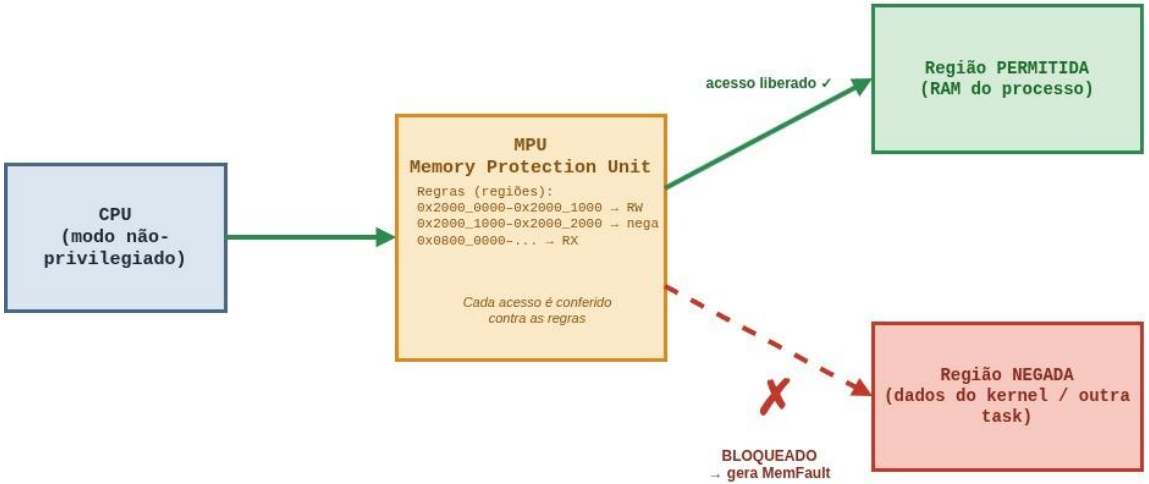
Sem barreiras de hardware: um bug em qualquer Task pode corromper a memória de outra — ou derrubar o sistema inteiro.

Níveis de Privilégio no Microcontrolador



MOUSER ELECTRONICS

A MPU: um Guarda entre a CPU e a Memória

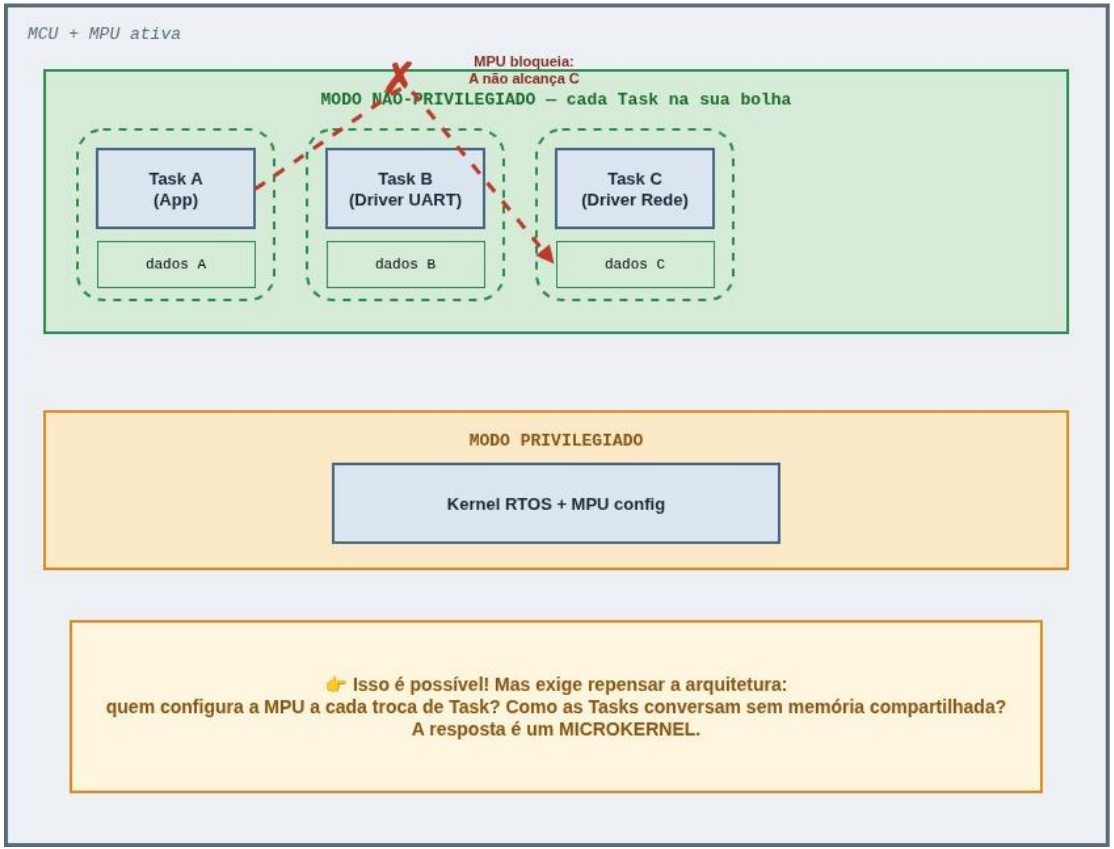


Em modo não-privilegiado, TODO acesso à memória passa pela MPU.
Se o endereço não estiver numa região permitida, o hardware bloqueia e dispara uma exceção — antes de qualquer dano acontecer.

RISC-V usa PMP e TriCore usa ranges DPR/CPR — mesma ideia: o hardware verifica cada acesso.

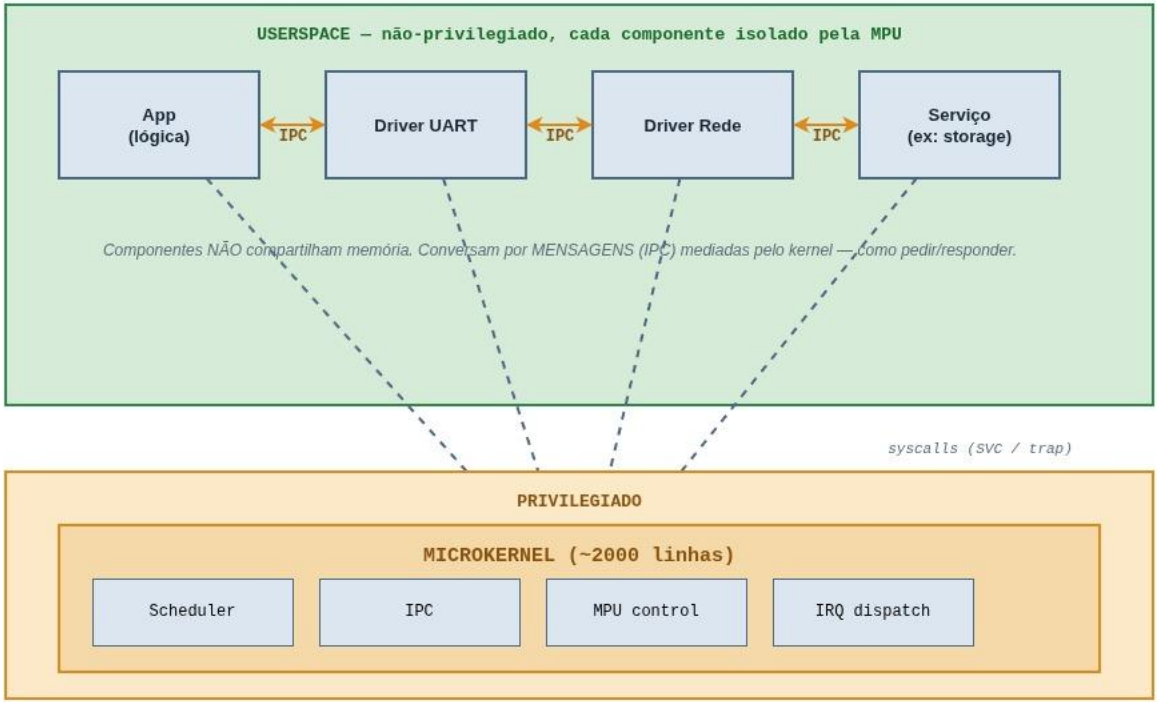
EMBARCADOS

E se o RTOS isolasse cada Task por hardware?



A Filosofia do Microkernel

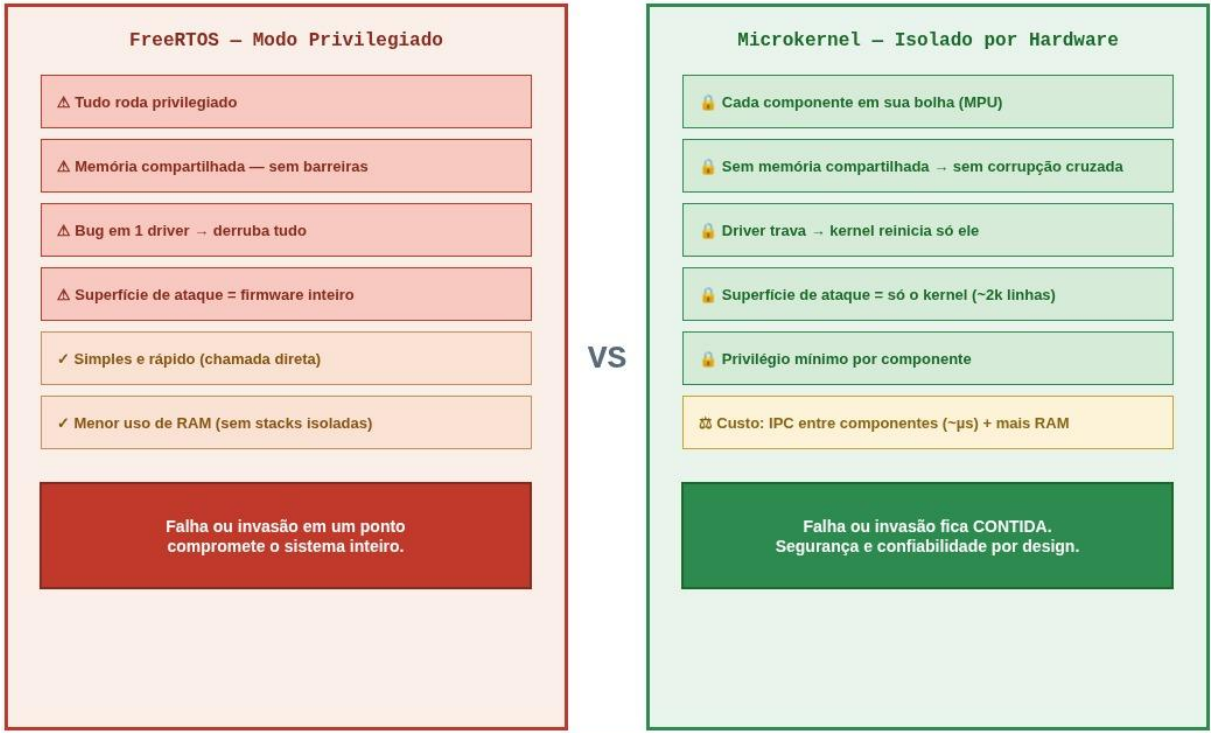
Tudo que puder rodar em userspace, roda em userspace. O kernel fica mínimo.



Se um driver travar, o kernel apenas o reinicia — o resto do sistema nem percebe. O kernel privilegiado é tão pequeno que é fácil de auditar e confiar.

MOUSER ELECTRONICS

FreeRTOS Privilegiado vs Microkernel Isolado



🛡️ O grande ganho: SEGURANÇA. Isolar por hardware significa que um bug ou um invasor não escapa da sua bolha — essencial para sistemas críticos (automotivo, médico, industrial).

EMBARCADOS

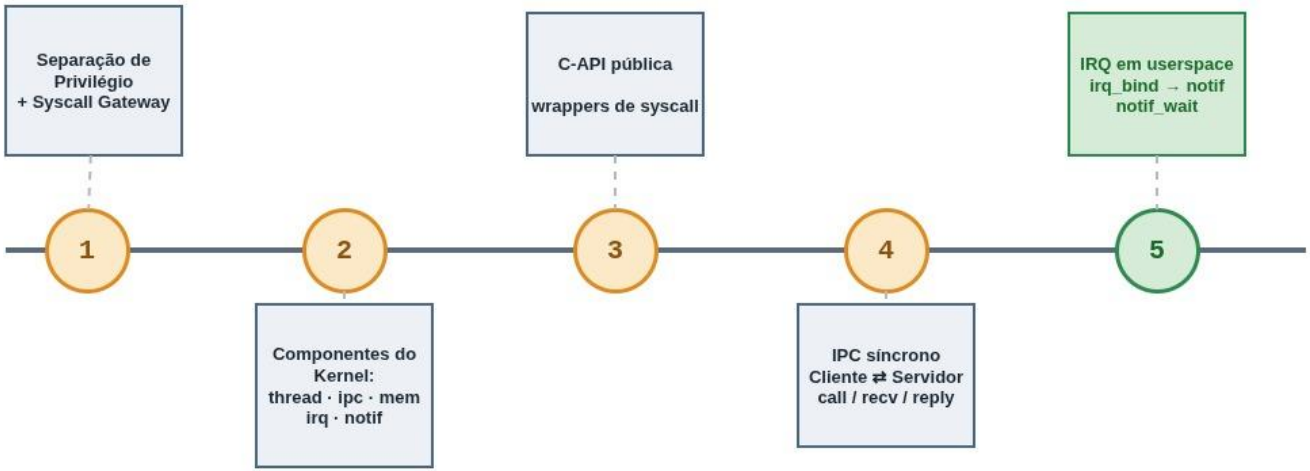
MOUSER ELECTRONICS

Parte II : O dia em que decidi escrever meu microkernel

EMBARCADOS

ulmk — A Jornada de Construção do Microkernel

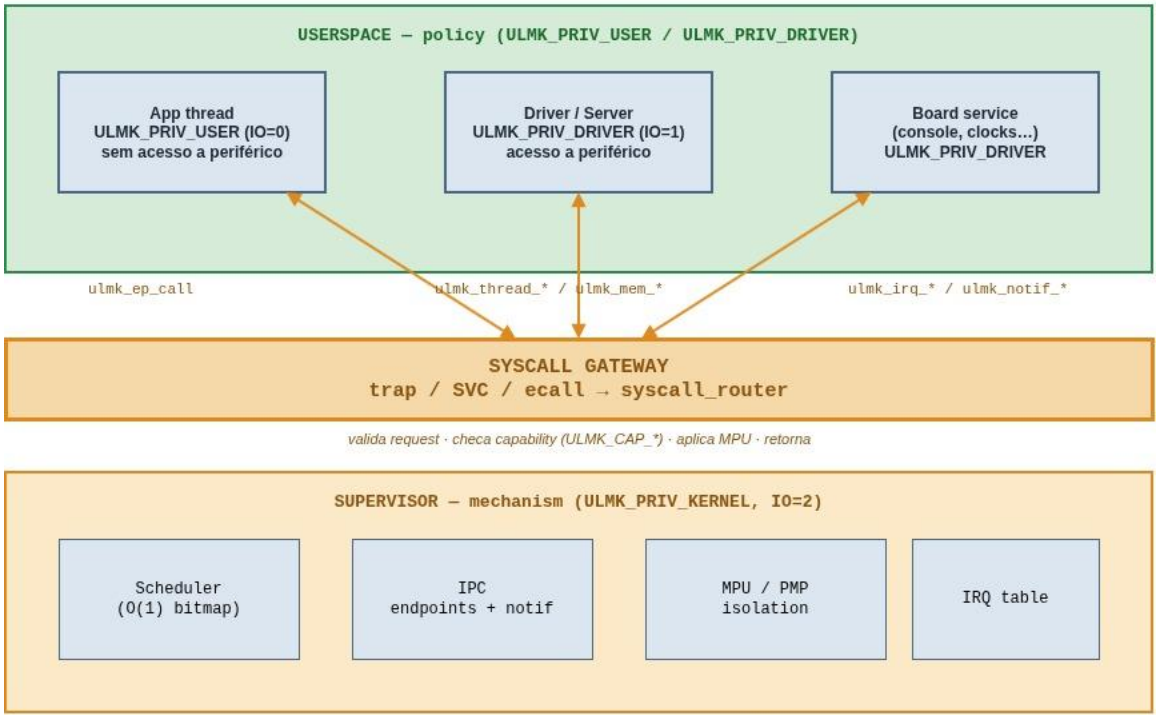
seL4-inspired · isolamento por MPU/PMP a nível de thread · policy em userspace



ARM Cortex-M (v7-M / v8-M) · RISC-V RV32IMAC · TriCore TC2xx/TC3xx — mesma API, isolamento por hardware

Separação de Privilégio: o Syscall Gateway

Toda operação privilegiada cruza um único portão — o kernel valida e aplica as barreiras da MPU.

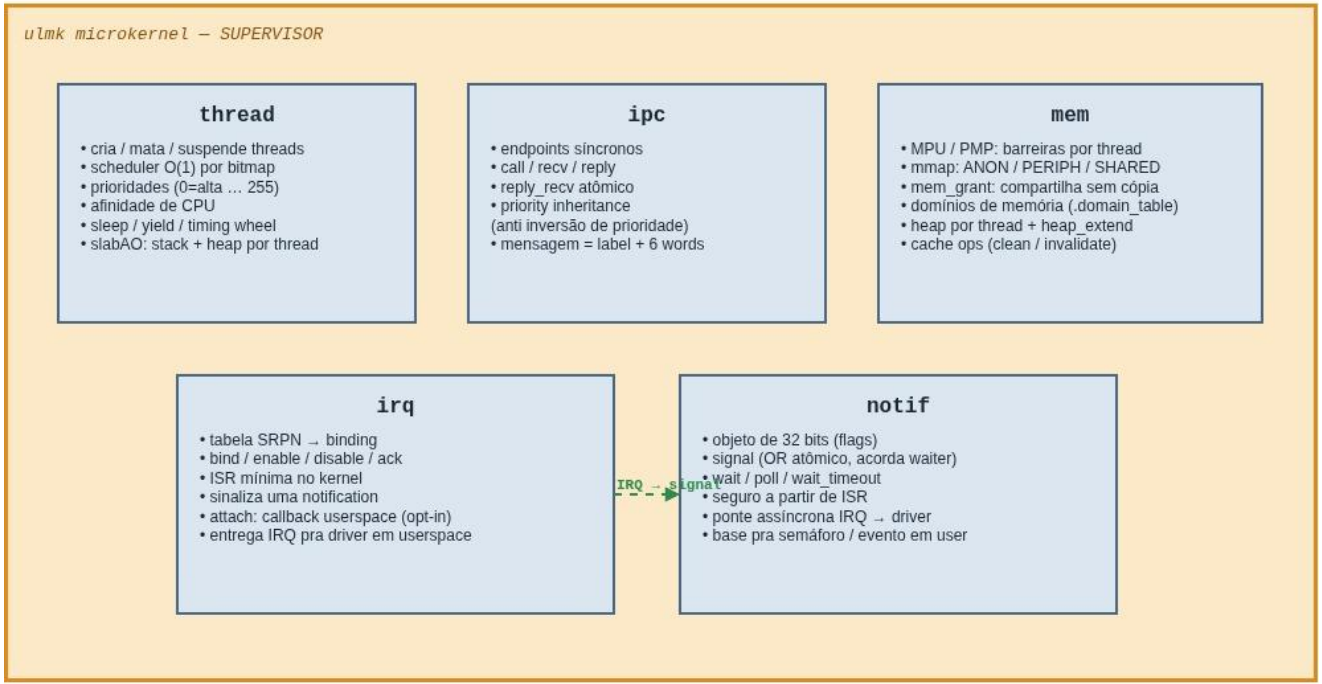


Threads nunca chamam umas às outras diretamente. Tudo é mensagem via kernel. Uma ELF única; a MPU mantém cada thread no seu domínio de dados.

PSW.IO no TriCore · Machine/User no RISC-V (PMP) · Privileged/Unprivileged no ARM Cortex-M

Componentes do Microkernel e Suas Funções

O kernel é só mecanismo — cinco subsistemas pequenos, auditáveis.



~2000 linhas no total. Tudo que pode ser policy (drivers, ordem de boot, topologia de serviços) vive em userspace.

MOUSER ELECTRONICS

C-API pública —

Cada função é um wrapper static inline que dispara o trap; o syscall_router despacha no kernel.

Thread & Scheduling

```
ulmk_thread_create(attr) → tid
ulmk_thread_kill(tid)
ulmk_thread_suspend/resume(tid)
ulmk_thread_self() → tid
ulmk_thread_priority_set/get(tid,p)
ulmk_thread_yield()
ulmk_sleep_ms(ms) / sleep_cancel(tid)
ulmk_thread_exit() [noreturn]
ulmk_cpu_id()
```

IPC — Endpoints (síncrono)

```
ulmk_ep_create() → ep
ulmk_ep_call(ep, msg)
ulmk_ep_call_timeout(ep, msg, ms)
ulmk_ep_recv(ep, &msg, &sender)
ulmk_ep_reply(sender, reply)
ulmk_ep_reply_recv(ep, s, rep, …)
ulmk_ep_recv_or_notif(ep, n, mask…)
ulmk_ep_grant(ep, target)
ulmk_ep_destroy(ep)
```

Notifications (assíncrono)

```
ulmk_notif_create() → n
ulmk_notif_signal(n, bits)
ulmk_notif_poll(n, mask)
ulmk_notif_wait(n, mask, &bits)
ulmk_notif_wait_timeout(…)
ulmk_notif_destroy(n)
```

Memory (requer capability)

```
ulmk_mem_map(hint, sz, perms, flags)
flags: ANON | PERIPH | SHARED
ulmk_mem_unmap(addr, sz)
ulmk_mem_grant(addr, sz, tid, perms)
ulmk_get_thread_heap(&info)
ulmk_heap_extend(sz)
ulmk_dcache_clean / invalidate(…)
```

IRQ — ULMK_PRIV_DRIVER + ULMK_CAP_IRQ

```
ulmk_irq_bind(srpn, notif, bit)
ulmk_irq_bind_hw(srpn, n, bit, src)
ulmk_irq_enable/disable(srpn)
ulmk_irq_ack(srpn)
ulmk_irq_attach(srpn, fn, data)
ulmk_irq_attach_hw(…)
ulmk_irq_detach(srpn)
```

Capabilities

```
ulmk_cap_grant(tid, caps)

ULMK_CAP_SPAWN
ULMK_CAP_KILL
ULMK_CAP_IRQ
ULMK_CAP_MAP_PERIPH
ULMK_CAP_MAP_SHARED
ULMK_CAP_GRANT_CAP
```

`ulmk_msg_t = { uint32_t label; uint32_t words[6]; } typedef uintptr_t ulmk_tid_t / ulmk_ep_t / ulmk_notif_t`

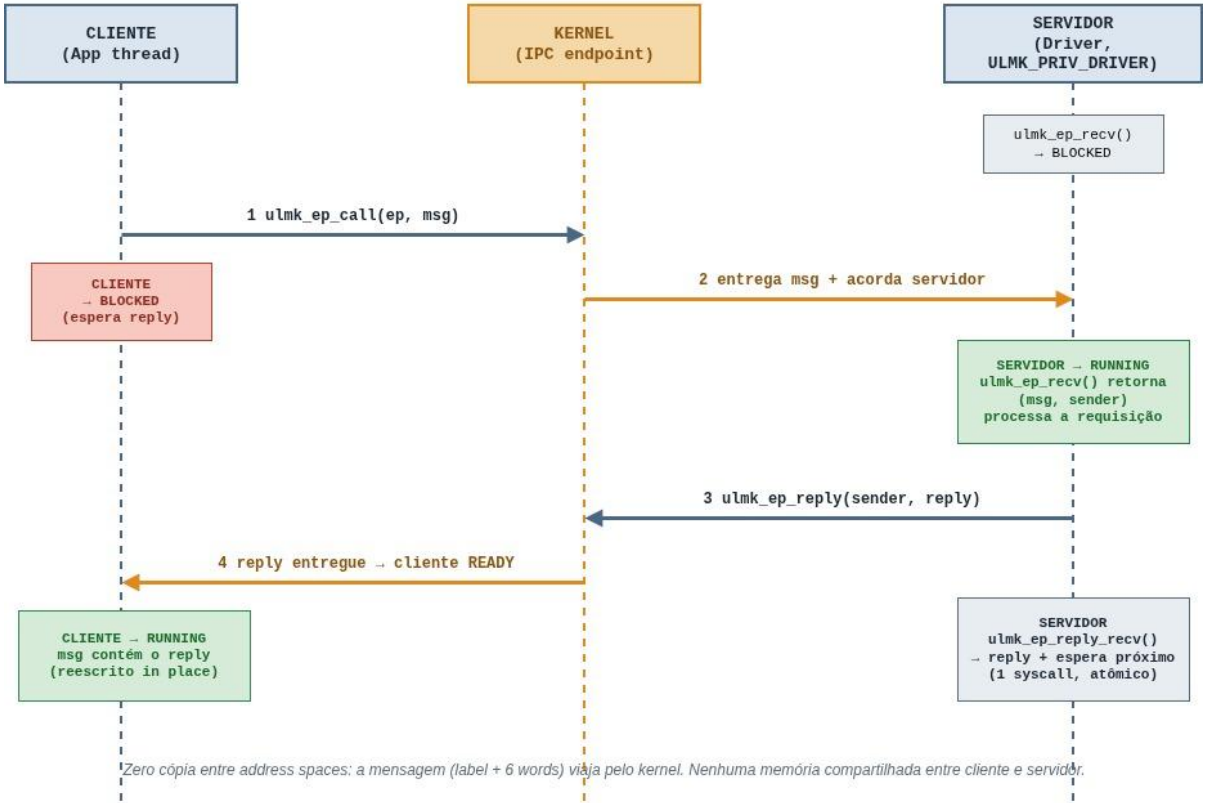
Boot: `void ulmk_root_thread(const ulmk_boot_info_t *info)` — criado pelo kernel em ULMK_PRIV_DRIVER com ULMK_CAP_ALL.

EMBARCADOS

MOUSER ELECTRONICS

IPC Síncrono: Padrão Cliente-Servidor

O cliente bloqueia até o servidor responder — priority inheritance evita inversão de prioridade.



Zero cópia entre address spaces: a mensagem (label + 6 words) viaja pelo kernel. Nenhuma memória compartilhada entre cliente e servidor.

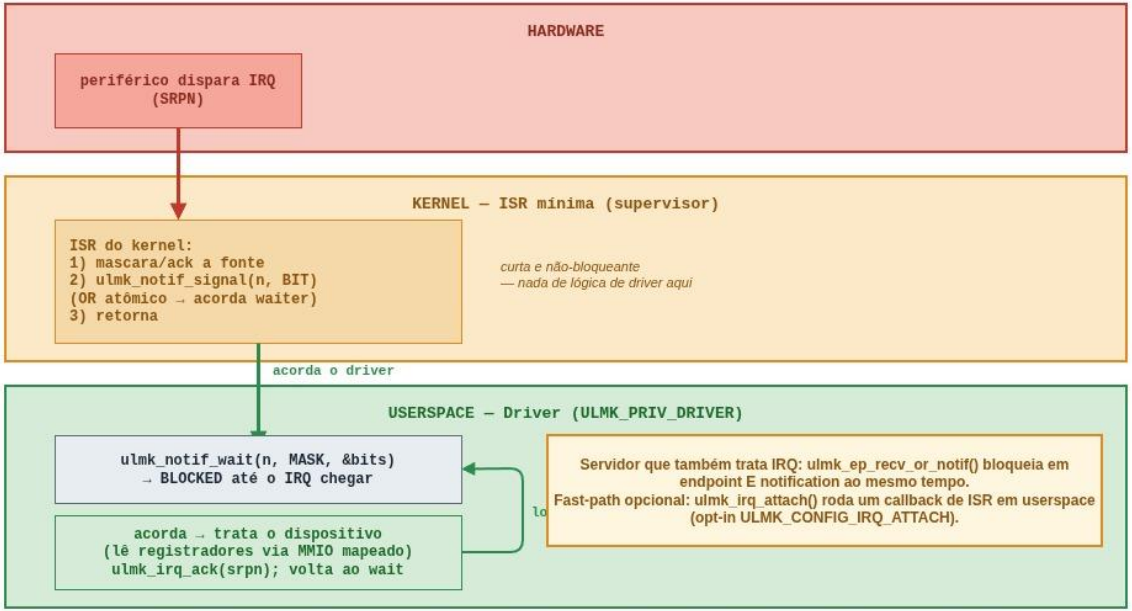
EMBARCADOS

Interrupções em Userspace: irq_bind + notif_wait

O hardware interrompe o kernel; o kernel só sinaliza uma notification; o driver em userspace acorda e trata.

SETUP (uma vez, no driver — ULMK_PRIV_DRIVER + ULMK_CAP_IRQ)

```
n = ulmk_notif_create();
ulmk_irq_bind(srpn, n, BIT); // liga a fonte de IRQ ao bit da notif
ulmk_irq_enable(srpn); // habilita entrega
```



srpn = source/priority no TriCore · IRQ line no ARM (NVIC) · IRQ id no RISC-V (PLIC). A tabela SRPN→notif tem ULMK_CONFIG_MAX_IRQ_BINDINGS slots.

MOUSER ELECTRONICS

Parte III : Chega de slides, Show me the code!

EMBARCADOS

MOUSER ELECTRONICS

Dúvidas?

EMBARCADOS

Links Uteis:

Links Uteis:

- **ulmk - Documentos:** https://github.com/uLipe/ulmk_microkernel/tree/main/docs
- **ulmk - microkernel:** https://github.com/uLipe/ulmk_microkernel
- **ulmk - Apps:** https://github.com/uLipe/ulmk_apps
- **ulmk - Boards:** https://github.com/uLipe/ulmk_boards
- **Meu contato:** https://github.com/uLipe
- **Ou:** ryukokki.felipe@gmail.com

OBRIGADO!

www.embarcados.com.br

linkedin.com/embarcados

@portalembarcados

youtube/Embarcados TV

EMBARCADOS

Patrocinado por

MOUSER ELECTRONICS