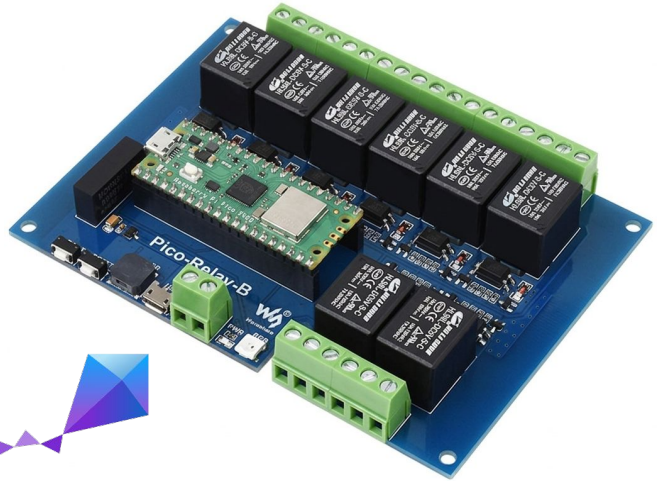
MOUSER ELECTRONICS

WEBINAR
EMBARCADOS



Zephyr RTOS construindo uma aplicação IoT completa

JORGE GUZMAN
Fundador da **GZM**
Embedded Systems

EMBARCADOS

Patrocinado por

MOUSER ELECTRONICS

MOUSER ELECTRONICS

GZM EMBEDDED SYSTEMS

Jorge Guzman

Fundador da GZM Embedded Systems

Especialista sênior em sistemas embarcados com mais de **13 anos de experiência**. Atendendo empresas no desenvolvimento de firmwares para IoT e indústria.

Zephyr NuttX FreeRTOS

Web
gzm-emb.com

LinkedIn
/in/eng-jorge-guzman

EMBARCADOS

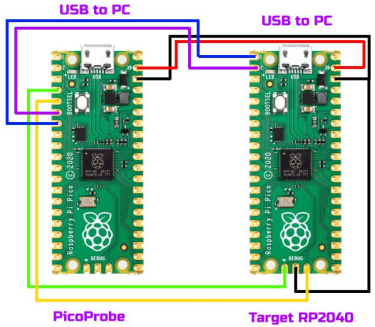
Agenda

- Hardware e ferramentas
- Stack Zephyr na Pico W
- Arquitetura da aplicação
- Testes e demo ao vivo

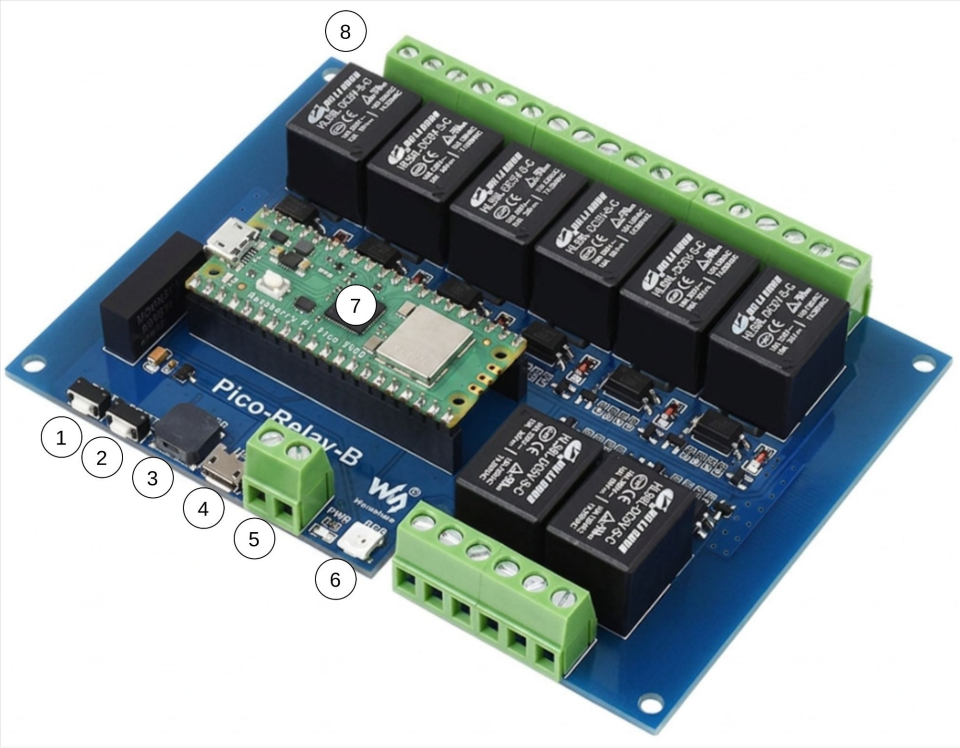
Esta e a continuação do **Zephyr RTOS: primeiros passos** de 2025

Material usado

- Software
  - Zephyr Project (RTOS) v4.4.0
  - Zephyr SDK (toolchain) v1.0.1
  - west & CMake
- Hardware
  - Pico-Relay-B (módulo de relé industrial)
  - Raspberry Pi Pico W (devkit)
- Debug
  - Raspberry Pi Debug Probe
  - Interface SWD - CMSIS-DAP
  - Open source, baseado em RP2040



WAVESHARE - PICO-RELAY-B



- 1 - Tecla Reset
- 2 - Tecla de Boot
  - usar: **zephyr.uf2**
- 3 - Buzzer Passivo
- 4 - Porta USB
- 5 - ⚠️ Alimentação 5V
- 6 - Led RGB WS2812
- 7 - Pi Pico W
  - **MCU RP2040**
  - **Dual-Core**
  - **264KB** de SRAM
  - **2MB** de Flash
  - **26** GPIOs expostos
  - Modulo Wifi AIROC CYW43
- 8 - 8 x Reles

Zephyr RTOS

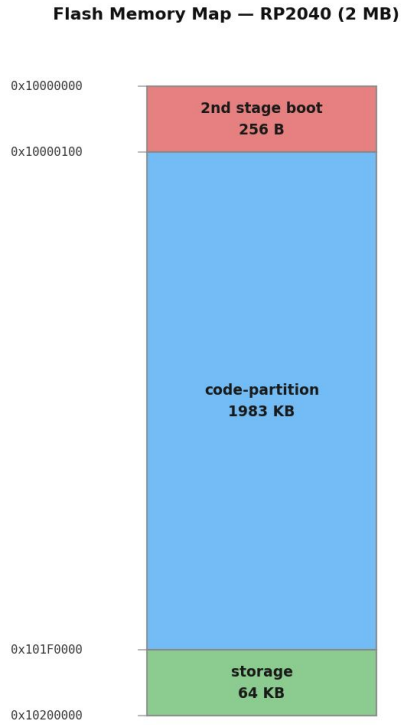
- Sistema operacional em tempo real open source
- Suporte a múltiplas arquiteturas e centenas de placas
- Uso de **Device Tree (DTS)** para descricao de hardware
- **Zephyr SDK**
  - GCC multi-arch (ARM, RISC-V, Xtensa, x86, ARC...)
  - OpenOCD: debug via SWD/JTAG
  - QEMU: emuladores para CI sem hardware
- Build system baseado em **cmake** e **west**
  - west init
  - west update
  - west flash
  - west debug
  - west blobs
  - west twister

Bibliotecas Usadas

- **Zephyr** (out of the box)
  - **Network:** WiFi (AIROC CYW43), DHCP, mDNS, SNTP
  - **HTTP:** HTTP Server (static + dynamic resources)
  - **Storage:** LittleFS (filesystem em flash)
  - **Periféricos:** GPIO, LED Strip (WS2812), PWM, RTC
  - **Diagnóstico:** Shell via USB CDC + Telnet
  - **Build:** **generate_inc_file_for_target** (gzip em compile time)
- **Customizadas (lib/gzm/)** - reutilizáveis entre projetos
  - **wifi_mgr:** bring-up + queue de eventos (CONNECTED, IP_OBTAINED, ...)
  - **rtc_mgr:** wrapper RTC com timezone (UTC <-> local)
  - **fs_mgr:** wrapper LittleFS (save/read/list/stats)
  - **ringtone:** player RTTTL via PWM (thread dedicada)

Flash NOR W25Q16JV - partições

```
&flash0 {
    partitions {
        compatible = "fixed-partitions";
        #address-cells = <1>;
        #size-cells = <1>;
        second_stage_bootloader: partition@0 {
            label = "second_stage_bootloader";
            reg = <0x00000000 0x100>;
            read-only;
        };
        code_partition: partition@100 {
            label = "code-partition";
            reg = <0x100 0x1EFF00>;
            read-only;
        };
        storage_partition: partition@1f0000 {
            label = "storage";
            reg = <0x1F0000 0x10000>;
        };
    };
};
```



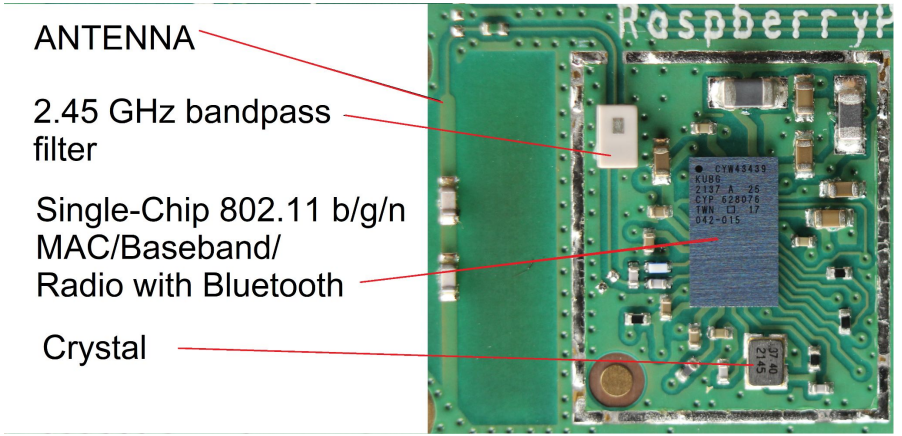
Conectividade: CYW43439

Conexão com o módulo WiFi via SPI por PIO(Programmable I/O)

- **GPIO29** OP/IP wireless SPI CLK
- **GPIO25** OP wireless SPI CS
- **GPIO24** OP/IP wireless SPI data/IRQ
- **GPIO23** OP wireless power on signal

prj.conf

```
CONFIG_WIFI=y
CONFIG_WIFI_AIROC=y
CONFIG_CYW43439=y
CONFIG_NET_L2_WIFI_MGMT=y
```



Driver WiFi: Zephyr

```
pio0_spi0: pio0_spi0 {
        compatible = "raspberrypi,pico-spi-pio";
        clocks = <&clocks RPI_PICO_CLKID_CLK_SYS>;
        #address-cells = <1>;
        #size-cells = <0>;
        cs-gpios = <&gpio0 25 GPIO_ACTIVE_LOW>;
        clk-gpios = <&gpio0 29 GPIO_ACTIVE_HIGH>;
        sio-gpios = <&gpio0 24 GPIO_ACTIVE_HIGH>;
        pinctrl-0 = <&pio0_spi0_default>;
        pinctrl-names = "default";
        status = "okay";

        airoc-wifi@0 {
            compatible = "infineon,airoc-wifi";
            reg = <0>;
            wifi-reg-on-gpios = <&gpio0 23 GPIO_ACTIVE_HIGH>;
            bus-select-gpios = <&gpio0 24 GPIO_ACTIVE_HIGH>;
            wifi-host-wake-gpios = <&gpio0 24 PIO_ACTIVE_HIGH>;
            spi-max-frequency = <10000000>;
            spi-half-duplex;
            spi-data-irq-shared;
            pinctrl-0 = <&airoc_wifi_default>;
            pinctrl-1 = <&airoc_wifi_host_wake>;
            pinctrl-names = "default", "host_wake";
            status = "okay";
        };
    };
```

zephyr/drivers/wifi/infineon/

- ├─ **airoc_wifi.c** ← driver principal (probe, init, scan, connect)
- ├─ **airoc_whd_hal_spi.c** ← HAL para barramento SPI (usado pelo Pico W)
- ├─ **airoc_whd_hal_sdio.c** ← HAL para barramento SDIO
- └─ **airoc_whd_hal_common.c** ← HAL compartilhado (IRQ, sleep, malloc)

zephyr/dts/bindings/wifi/infineon,airoc-wifi.yaml

└─ arquivo de binding YAML que as propriedades aceitas

⚠️ BLE não disponível: driver Zephyr para o CYW43439 implementa somente WiFi.

Notas musicais - Frequência

12 notas cromáticas em 4 oitavas. RTTTL = símbolo + oitava (ex: a4 = Lá oitava 4 = 440 Hz)

Nota	Símbolo	f (Hz) oitava 4	f (Hz) oitava 5	f (Hz) oitava 6	f (Hz) oitava 7
Dó	c	261.62	523.25	1046.50	2093.00
Dó#/Réb	c#	277.19	554.37	1108.74	2217.48
Ré	d	293.67	587.33	1174.66	2349.32
Ré#/Mib	d#	311.12	622.25	1244.50	2489.00
Mi	e	329.63	659.26	1318.52	2637.04
Fá	f	349.23	698.46	1396.92	2793.84
Fá#/Solb	f#	370.00	739.99	1479.98	2959.96
Sol	g	392.00	783.99	1567.98	3135.96
Sol#/Láb	g#	415.31	830.61	1661.22	3322.44
Lá	**a**	**440.00**	**880.00**	**1760.00**	**3520.00**
Lá#/Sib	a#	466.17	932.33	1864.66	3729.32
Si	b	493.88	987.77	1975.54	3951.08

Linha destacada: Lá (a4) = 440 Hz, padrão internacional ISO 16 · f(oitava+1) = 2 × f(oitava)

Temperamento igual: f(n) = 440 × 2^((n-69)/12)

RTTTL (Ring Tone Text Transfer Language)

Formato texto criado pela Nokia para transferir ringtones a celulares.

Estrutura: 3 seções separadas por `:`

`nome:defaults:notas` ⟶ `fifth:d=4,o=5,b=63:8p,8g5,8g5,8g5,2d#5`

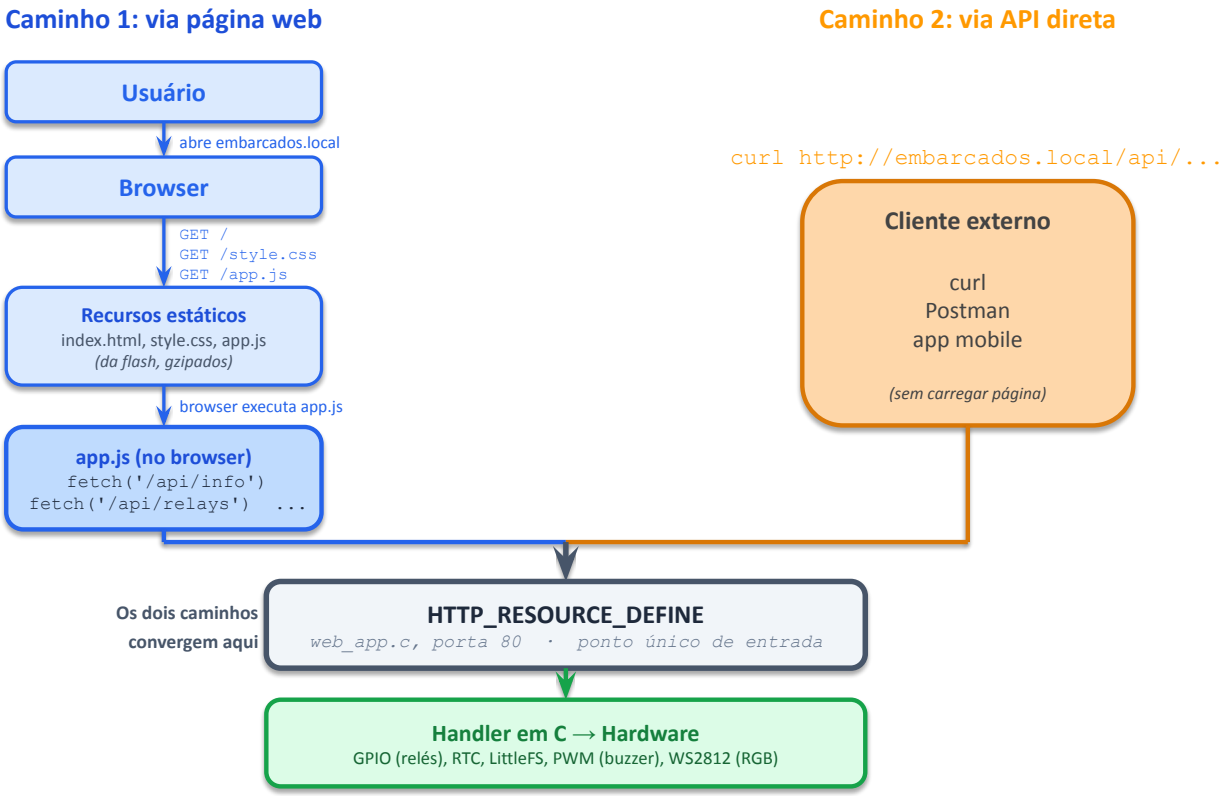
- **nome** — identificador da música, até 10 caracteres
- **defaults** — valores aplicados às notas que não os especificarem:
  - `d=` duração padrão. Valores válidos: `1, 2, 4, 8, 16, 32` (frações de compasso)
  - `o=` oitava padrão. Valores válidos: `4, 5, 6, 7`
  - `b=` andamento em BPM
- **notas** — separadas por vírgula, no formato `[duração][nota][oitava][.]`
  - Notas: `A–G`, sustenido `#`, pausa `P`
  - Duração e oitava são opcionais por nota (caem no default se omitidos)
  - `.` (ponto) prolonga a nota em 50%

Serviço Web

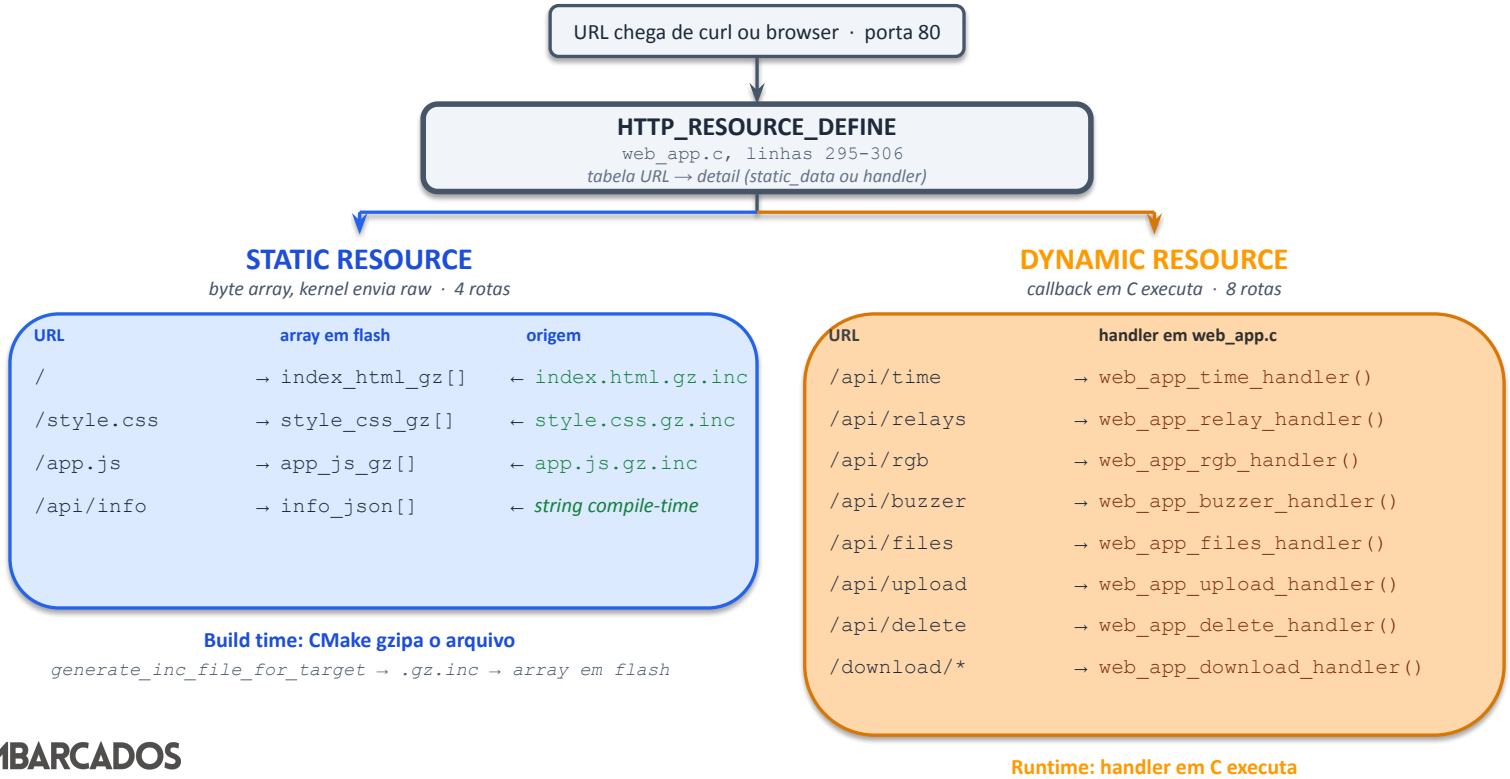
- HTTP Server: servidor web embarcado com recursos estáticos e API REST
  - **CONFIG_HTTP_SERVER=y**
  - **CONFIG_HTTP_SERVER_RESOURCE_WILDCARD=y**
- DHCP: obtenção automática de endereço IP na rede
  - **CONFIG_NET_DHCPV4=y**
- mDNS: acesso ao dispositivo por nome (ex: embarcados.local)
  - **CONFIG_MDNS_RESPONDER=y**
  - **CONFIG_NET_HOSTNAME="embarcados"**
- SNTP: sincronização de data e hora via rede
  - **CONFIG_SNTP=y**
- Shell remoto: terminal shell exposto via TCP
  - **CONFIG_SHELL_BACKEND_TELNET=y**
- Compressão em build
  - **generate_inc_file_for_target(... gzip)**

index.html(6.4 KB) → gzig→index.html.gz.inc (2.2 KB)
comprime dados convertendo bytes em hex literal
Ex: Original: 6546 bytes (6.4 KB)
Gzipado: 2221 bytes (2.2 KB)
Economia: 66.1% (sobra ~1/3 do tamanho)

HTTP - Rotas


Caminho 1: via página web
Usuário
abre embarcados.local
Browser
GET /
GET /style.css
GET /app.js
Recursos estáticos
index.html, style.css, app.js
(da flash, gzipados)
browser executa app.js
app.js (no browser)
fetch('/api/info')
fetch('/api/relays') ...
Caminho 2: via API direta
curl http://embarcados.local/api/...
Cliente externo
curl
Postman
app mobile
(sem carregar página)
Os dois caminhos convergem aqui
HTTP_RESOURCE_DEFINE
web_app.c, porta 80 · ponto único de entrada
Handler em C → Hardware
GPIO (relés), RTC, LittleFS, PWM (buzzer), WS2812 (RGB)

Recursos HTTP: para onde vai cada URL



Estrutura do projeto

Organizado como Zephyr Module, código de domínio em `lib/` reutilizável entre apps e testes.

```
project/
├── .vscode/ — build, flash, debug e testes
├── app/     — aplicação principal
│   └── www/ — página web
├── app_2/   — outra app que reusa libs/
├── boards/  — placa out-of-tree
├── lib/     — módulo Zephyr (reutilizável)
│   ├── common/
│   ├── fs_mgr/
│   ├── ringtone/
│   ├── wifi_mgr/
│   ├── zephyr/
│   └── Kconfig
└── README.md — Documentacao
```

REUTILIZAÇÃO

lib/ registrada como Zephyr Module

Apps apontam via **ZEPHYR_EXTRA_MODULES** e habilitam libs por Kconfig.

CADA LIB

CMakeLists + Kconfig + include + src

```
lib/<nome>/
├── CMakeLists.txt
├── Kconfig  # CONFIG_GZM_*=y
├── include/gzm/<nome>.h
└── src/<nome>.c
```

SHOW ME THE CODE

- DEMO
  - https://github.com/JorgeGzm/WebinarEmbarcados2026
  - Documentacao em **README.md**
  - Task runners
    - Build
    - Flash
    - Debug

Próximos passos

- **Qualidade de Código**
  - **cppcheck:** procura bugs como buffer overflow, null pointer dereference, memory leak
  - **lizard:** Identifica funções "monstro" com muitos if/while/switch para refactor
  - **flawfinder:** Detecta uso de funções inseguras (strcpy, gets, sprintf)
  - **fstack-usage:** uso de stack por função
- **Simuladores**
  - **Renode:** simulador de **sistema embarcado completo** (CPU + periféricos).
  - **QEMU:** Simulador genérico de CPU
- **Testes**
  - **ztest:** framework de testes unitário**s** do Zephyr roda com west twister
    - testes usando native_sim
    - testes direto no target
  - **coverage:** gera relatório HTML com linhas cobertas pelos testes
- **Robot Framework:**
  - Testes de validação da aplicação
  - Validar a aplicacao rodando como um todo
- **CI/CD**
  - **GitHub Actions:** roda twister a cada push
  -

OBRIGADO!

www.embarcados.com.br

linkedin.com/embarcados

@portalembarcados

youtube/Embarcados TV

EMBARCADOS

Patrocinado por

MOUSER ELECTRONICS

Contato

Jorge Guzman

GZM EMBEDDED SYSTEMS

- **Email**: jorge@gzm-emb.com
- **Linkedin**: linkedin.com/in/eng-jorge-guzman
- **Webpage**: gzm-emb.com

MOUSER ELECTRONICS

EMBARCADOS