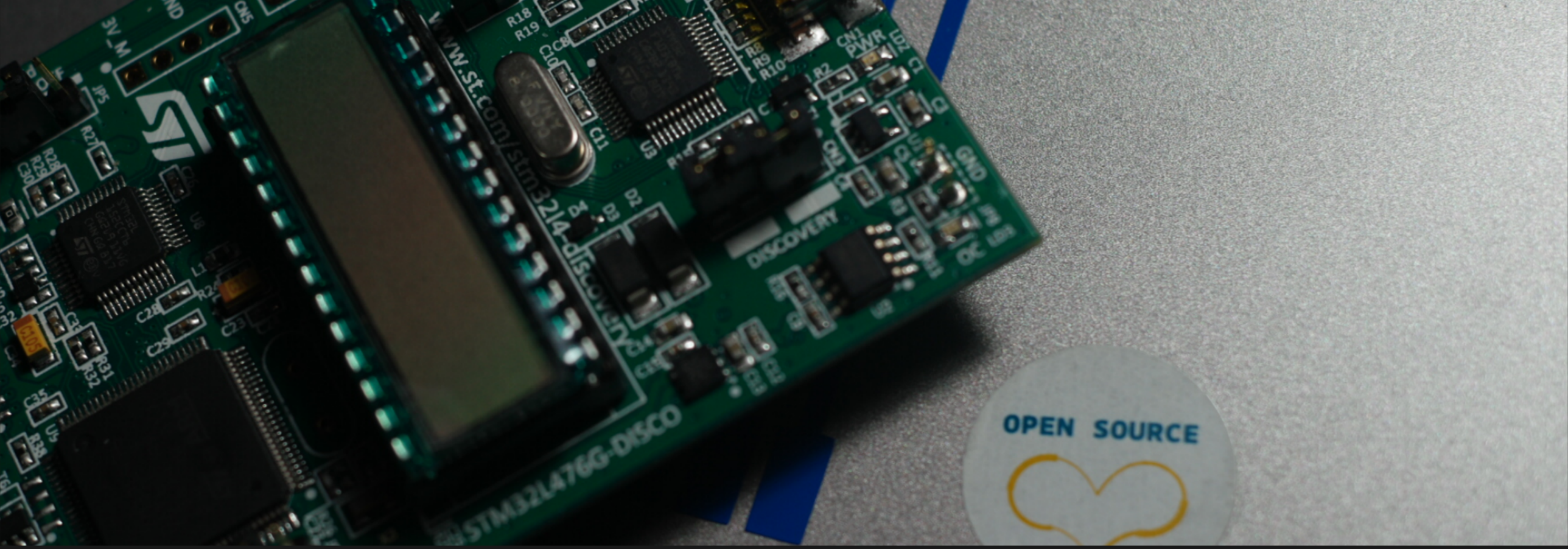


DEPURACIÓN EN GDB CON OPENOCD PARA SISTEMAS EMBEBIDOS

KIARA NAVARRO

@sophiekovalevsky


Debugger
GDB
Breakpoint
ST-Link
JLink
Watchpoint
SWD
OpenOCD
MemoryMap
JTAG
Stack Frame
GCC
BareMetal

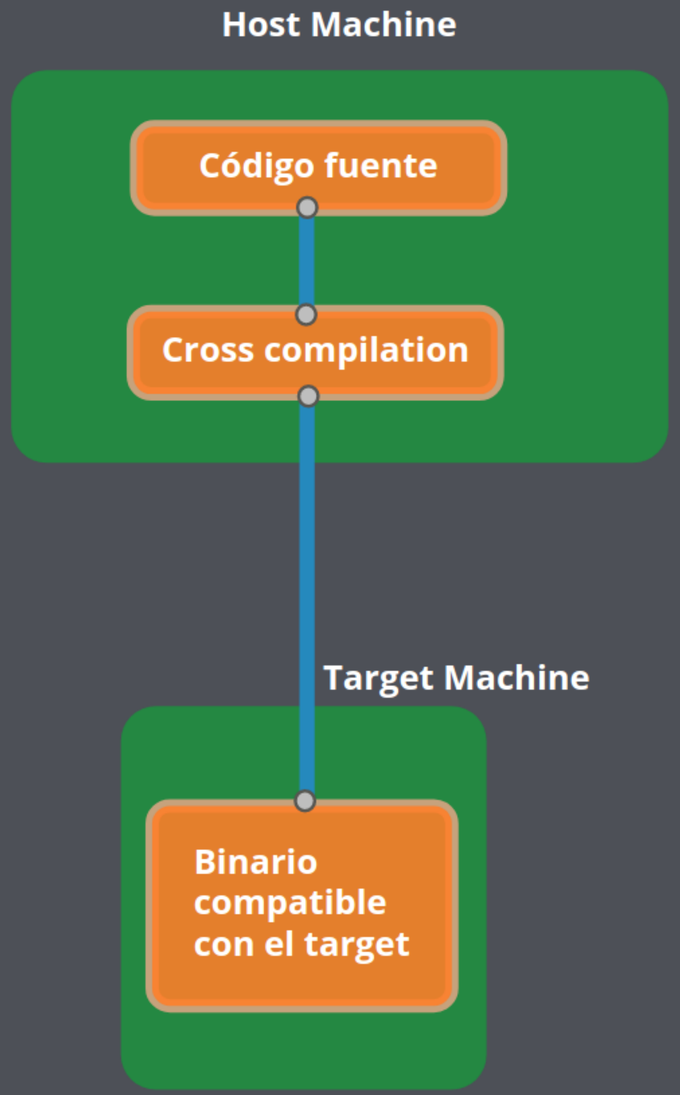
CROSS-COMPILING TOOLCHAIN

Toolchain de compilación:

- Programas que ayudan a tener como resultado final, archivos binarios a partir del código fuente.

Cross:

- Cuando el binario no se ejecuta en la plataforma donde se construyó.



COMPONENTES

- GNU Compiler Collection (GCC)
- GNU Binutils
- GDB (Nuestra ⭐ principal)
- Newlib

```
<arquitectura-del-core>-<sistema-operativo>-<tipo-de-abi>
```

- architectura: arm
- sistema operativo: no tiene, bare-metal
- EABI: compila con ARM Embedded ABI

```
gcc-arm-none-eabi-10-2020-q4-major/bin
├── arm-none-eabi-addr2line
├── arm-none-eabi-ar
├── arm-none-eabi-as (ensamblador)
├── arm-none-eabi-c++
├── arm-none-eabi-c++filt
├── arm-none-eabi-cpp
├── arm-none-eabi-elfedit
├── arm-none-eabi-g++
├── arm-none-eabi-gcc (orquestrador)
├── arm-none-eabi-gcc-10.2.1
├── arm-none-eabi-gcc-ar
├── arm-none-eabi-gcc-nm
├── arm-none-eabi-gcc-ranlib
├── arm-none-eabi-gcov
├── arm-none-eabi-gcov-dump
├── arm-none-eabi-gcov-tool
├── arm-none-eabi-gdb (client gdb)
├── arm-none-eabi-gdb-add-index
├── arm-none-eabi-gdb-add-index-py
├── arm-none-eabi-gdb-py
├── arm-none-eabi-gprof
├── arm-none-eabi-ld (linker)
├── arm-none-eabi-ld.bfd
├── arm-none-eabi-lto-dump
├── arm-none-eabi-nm
├── arm-none-eabi-objcopy
├── arm-none-eabi-objdump
├── arm-none-eabi-ranlib
├── arm-none-eabi-readelf
├── arm-none-eabi-size
├── arm-none-eabi-strings
└── arm-none-eabi-strip
```

¿QUÉ PERMITE LA TÉCNICA DE DEPURACIÓN? 🐝

- Inspecciona contenido en memoria FLASH y RAM (registros)
- Detiene el programa
- Ejecuta código paso a paso
- Permite colocar breakpoints y watchpoints
- Modifica variables en tiempo de ejecución

Créditos imagen: Editora NovoArt

SÍMBOLOS DE DEPURACIÓN

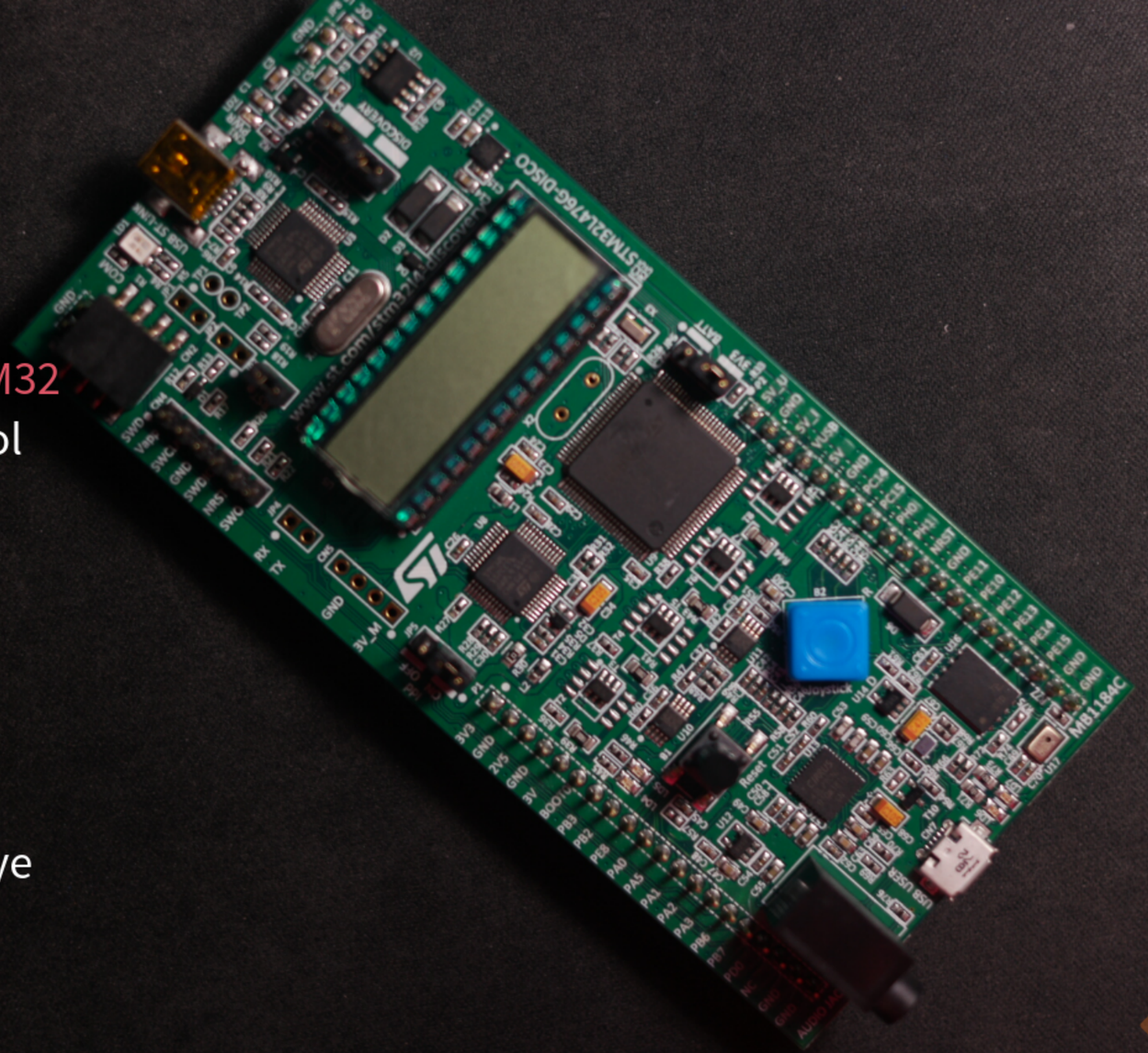
```
file DebugginOpenOcd.elf
arm-none-eabi-readelf -h DebugginOpenOcd.elf

ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x8001075
  Start of program headers:          52 (bytes into file)
  Start of section headers:          1203344 (bytes into file)
  Flags:                             0x5000400, Version5 EABI, hard-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         3
  Size of section headers:           40 (bytes)
  Number of section headers:         25
  Section header string table index: 24
```

- C flag, nivel de depuración:

```
-gNivel
```

HERRAMIENTAS

- Software
  - Código boilerplate
  - System Workbench for STM32
    - C/C++ Development Tool package
    - OpenOCD
    - GNU ARM C/C++ Cross Compiler
  - Python 3+
- Hardware
  - STM32L476G-DISCO (incluye debug probe ST-LINK)

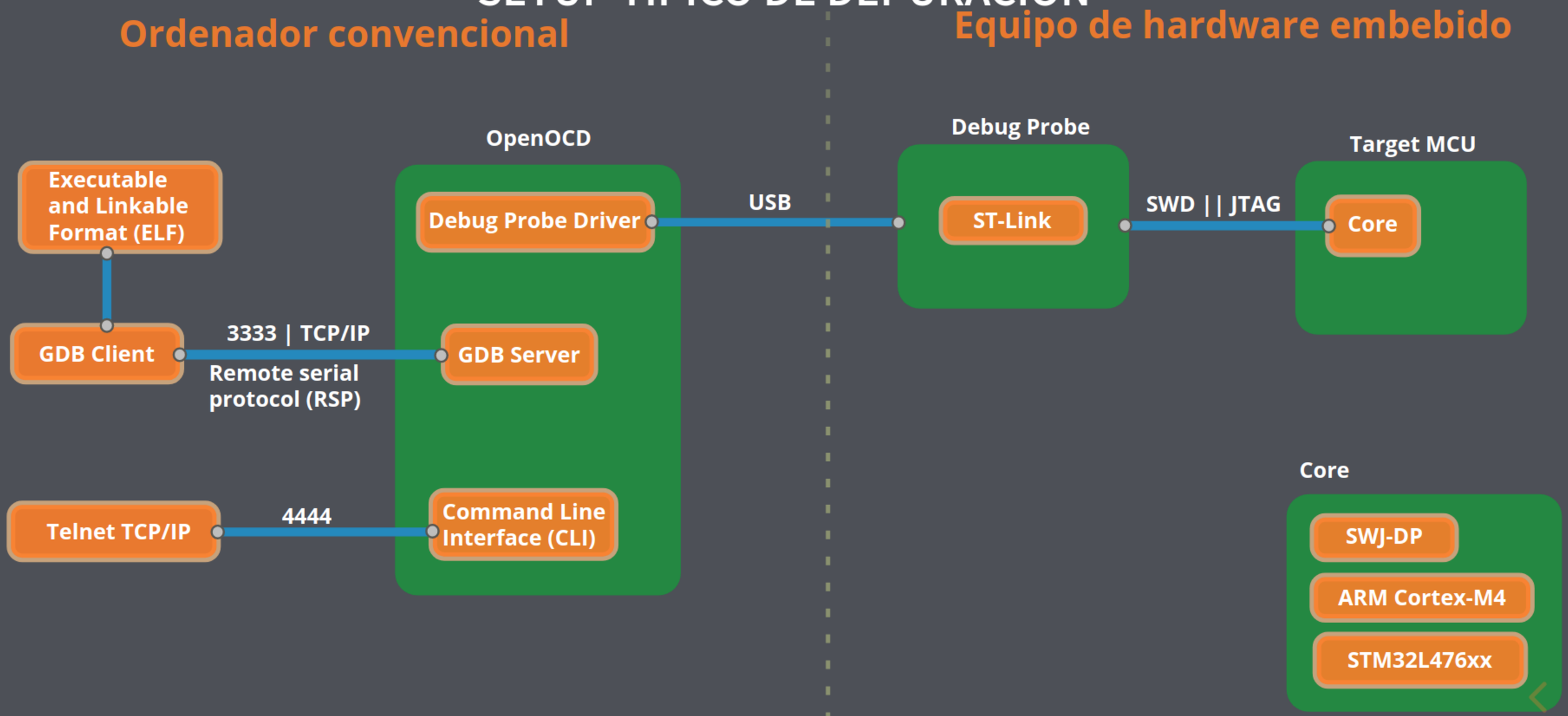


OPEN ON-CHIP DEBUGGER

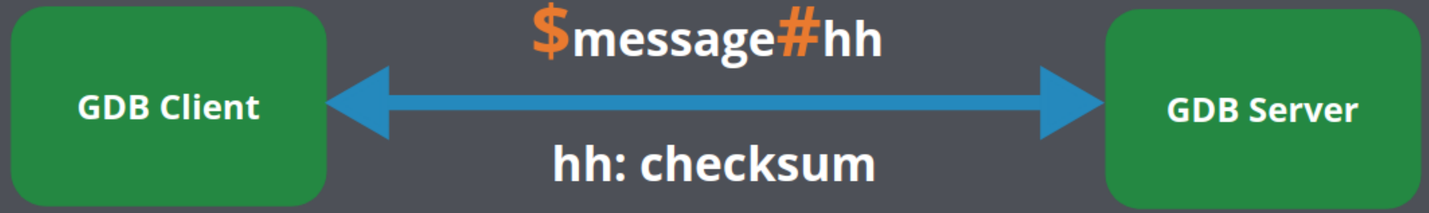
Herramienta de depuración para hardware embebido.

```
openocd -s /mi/propio/directorio/de/busqueda -f -<interface> -f <target>
ls -lh /usr/share/openocd/scripts/interface
ls -lh /usr/share/openocd/scripts/target
openocd -d -f interface/stlink-v2-1.cfg -f target/stm32l4x.cfg
```

SETUP TÍPICO DE DEPURACIÓN



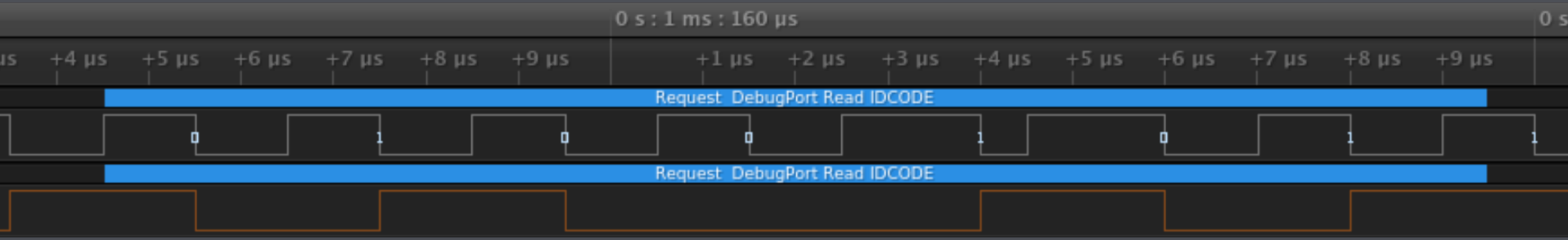
GDB REMOTE SERIAL PROTOCOL



Implementación

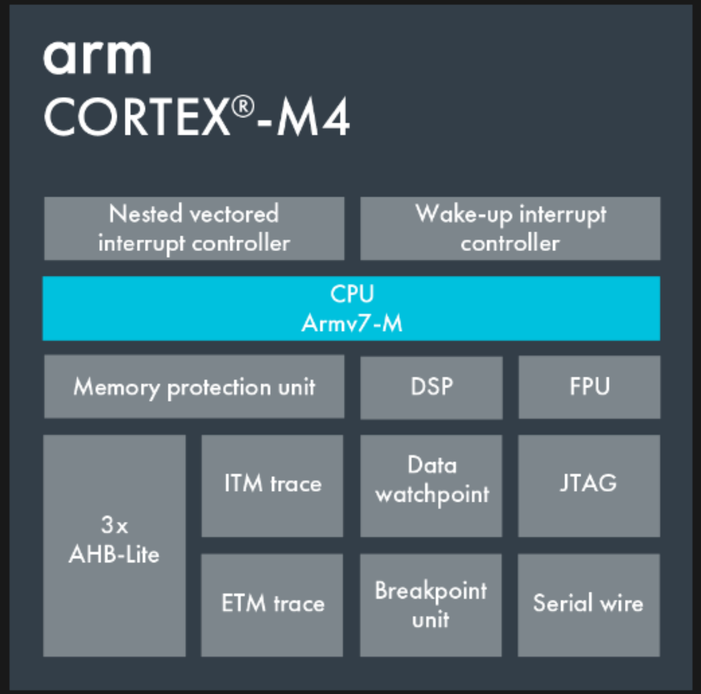
- GDB Server fake in 🐍

SERIAL WIRE DEBUG PROTOCOL



- Protocolo capa física
- JTAG por defecto

BREAKPOINTS Y WATCHPOINTS



BREAKPOINTS

```
(gdb) info breakpoints
(gdb) break <location>
(gdb) delete break
```

WATCHPOINTS

- No aplica a variables locales, únicamente globales y estáticas

```
(gdb) info watchpoints
```

STACK TRACE

```
(gdb) bt
(gdb) bt full
(gdb) frame <n>
(gdb) up
(gdb) down
(gdb) step (step into)
(gdb) next (step over)
```