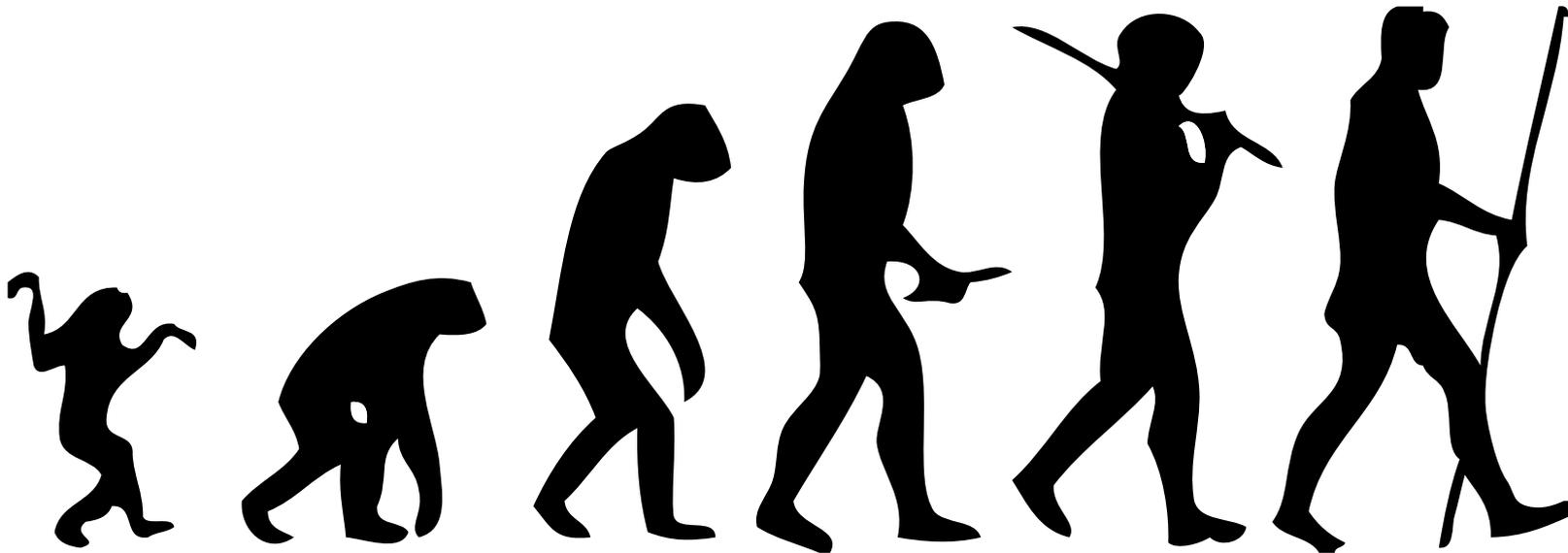




UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

DISCA

Un “Hola Mundo” en C para microcontroladores ARM Cortex-M



2015/06/15

Àngel Perles



armcortexm.blogs.upv.es



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



etsinf

DISCA

Contenido

- Objetivos
- El lenguaje C es lo adecuado
- Proyectos a partir de una plantilla
- “Hola Mundo” al servicio de depuración



Objetivos

- Ver algo funcionando
- Usar el compilador Keil para construir un proyecto
- Crear un proyecto a partir de plantilla



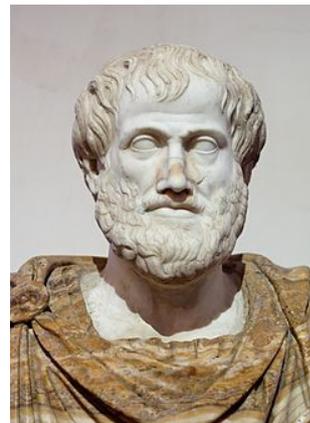
El lenguaje C es lo adecuado

- Esta arquitectura está pensada para desarrollo en lenguaje C
 - Usar ensamblador no tiene sentido (ni aquí ni en otros sitios)
 - (Un programador experto conoce ensamblador y le da ventaja respecto al resto de programadores)
- Hay otras opciones:
 - Pascal, Basic, C++, C#, java, Ada, ...
 - Son lenguajes marginales en el ámbito de los microcontroladores
- Como ingeniero, elige ...

Inglés

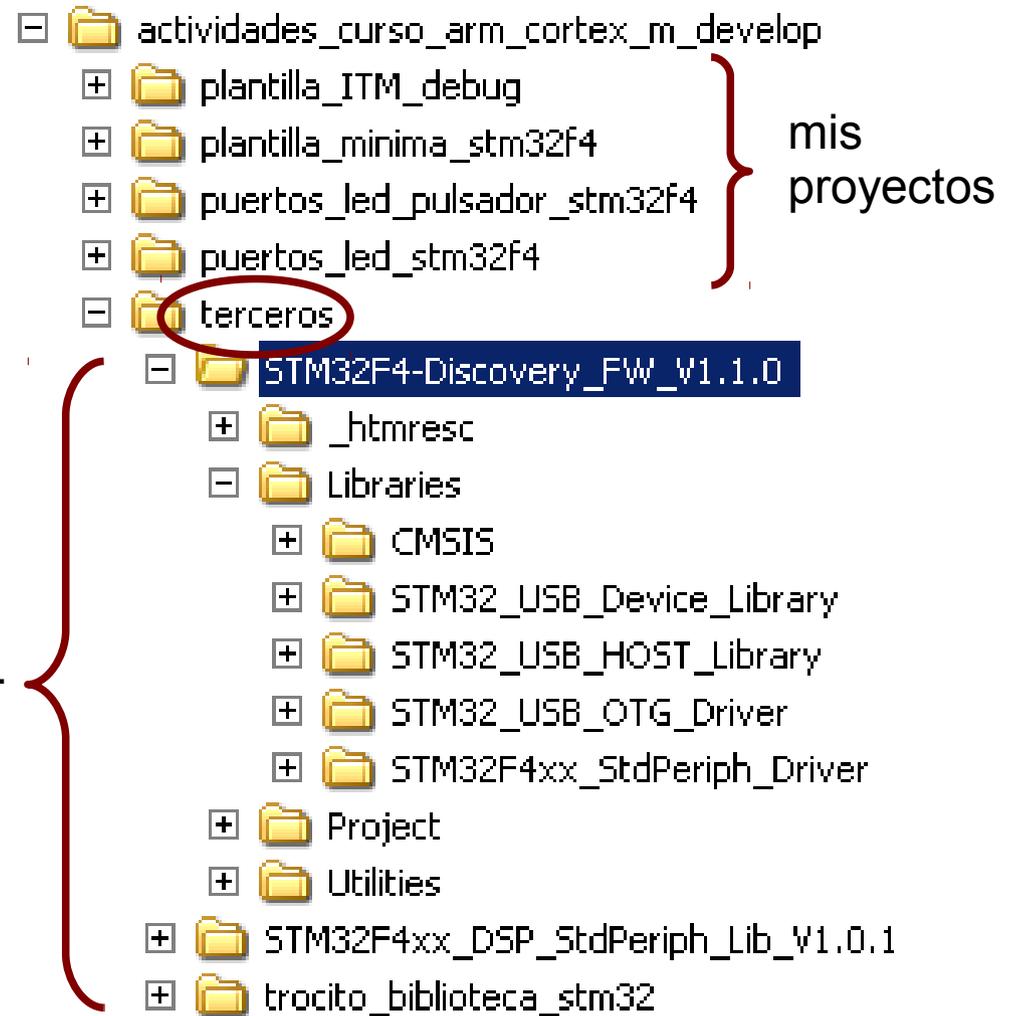


Griego



Proyectos a partir de una plantilla

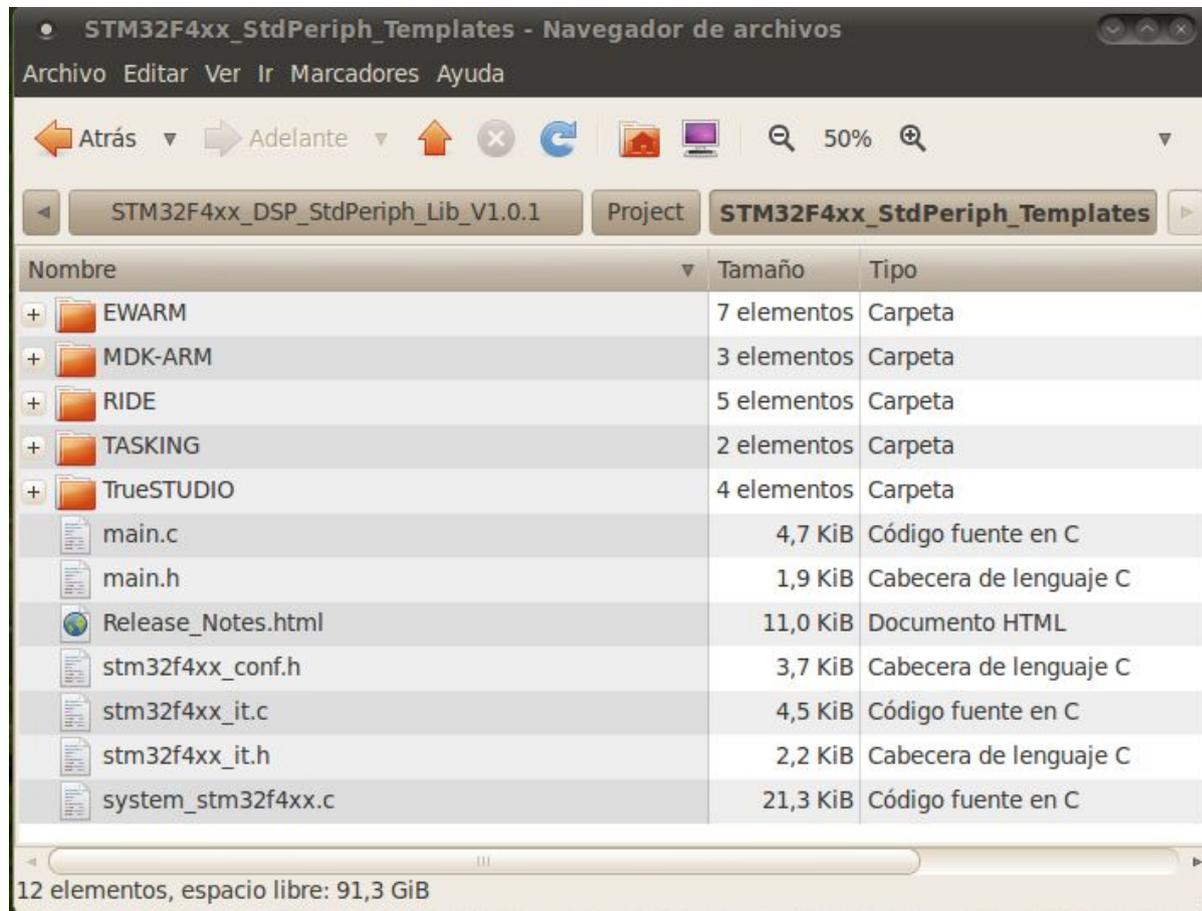
- Un proyecto ARM se debe apoyar en montones de bibliotecas
 - Hay que ser organizados
 - Una plantilla suele usar rutas preconfiguradas
 - En la rutas, evitar, espacios y símbolos raros



Proyectos a partir de una plantilla

- Y las bibliotecas

- Las proporcionan los fabricantes del chip, los entornos, ...
- St la proporciona con la “STM32F4 DSP and Standard peripheral library”



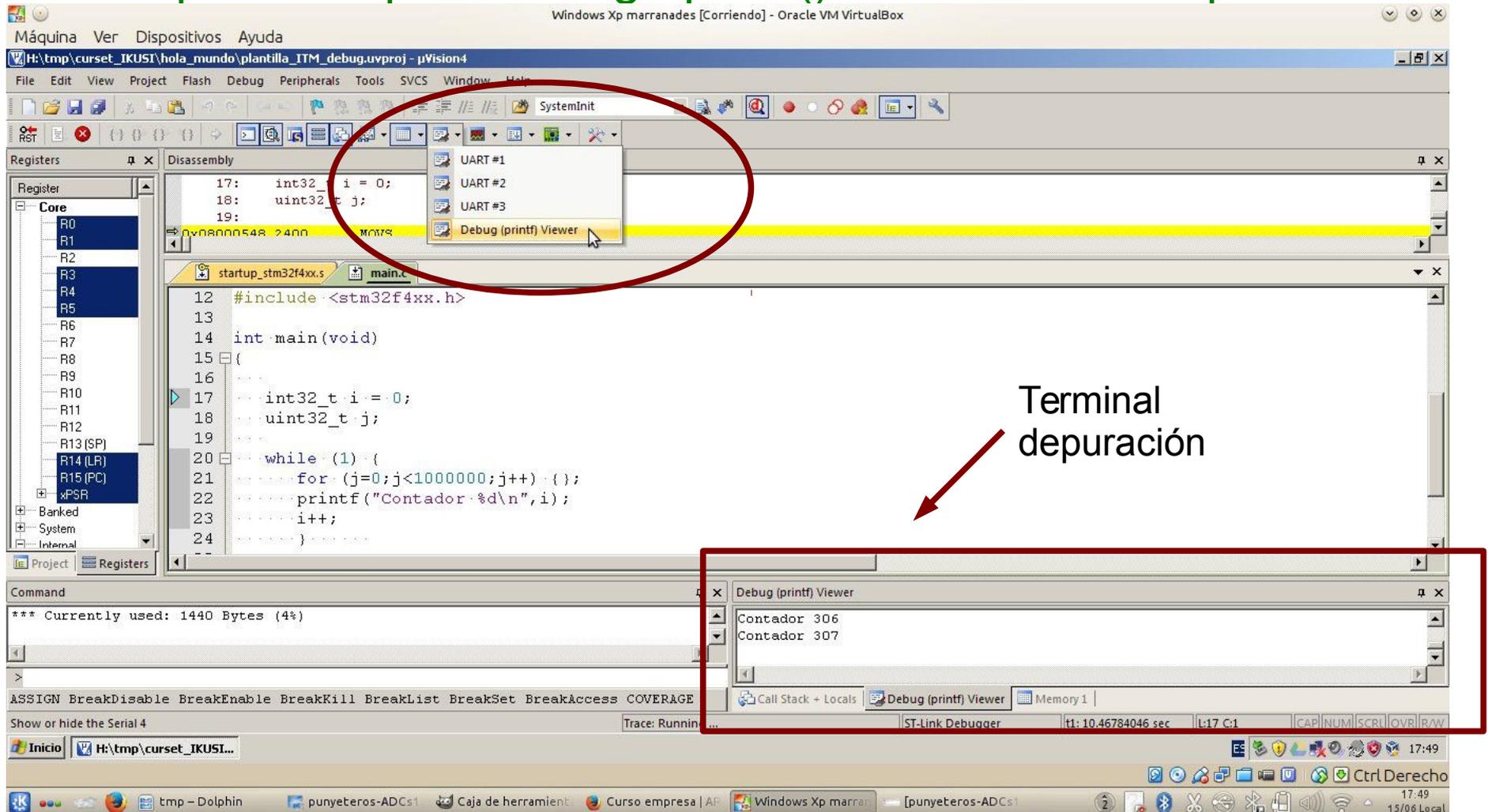
Proyectos a partir de una plantilla

- Hemos creado una plantilla para el curso
 - Localiza un sitio para trabajar. Por ejemplo C:\curso
- Actividad: Usar la plantilla
 - 1 - Descomprime la “plantilla con printf() ITM debug” en el directorio de trabajo
 - 2 - Descarga “STM32F4DISCOVERY board firmware package”
 - 3 - Descomprímelo en el subdirectorio “terceros”. NO USES EL DESCOMPRESOR POR DEFECTO DE WINDOWS, LO HACE MAL (usa 7zip, winzip, winrar, ... o similar)
 - 4 - Con Keil, abre el proyecto “plantilla_ITM_debug.uvproj”
 - 6 - Prueba a compilar



Proyectos a partir de una plantilla

- Para facilitar las pruebas, la plantilla utiliza una característica de depuración para redirigir printf() a un terminal especial



The screenshot shows the ST-Link Debugger interface. The main window displays the disassembly and source code of a program. The source code is as follows:

```
12 #include <stm32f4xx.h>
13
14 int main(void)
15 {
16     ...
17     int32_t i = 0;
18     uint32_t j;
19     ...
20     while (1) {
21         for (j=0; j<1000000; j++) { };
22         printf("Contador %d\n", i);
23         i++;
24     }
25 }
```

The 'Debug (printf) Viewer' window at the bottom shows the output of the printf statements:

```
Contador 306
Contador 307
```

A red circle highlights the 'Debug (printf) Viewer' option in the context menu, and a red arrow points to the 'Debug (printf) Viewer' window at the bottom, which is labeled 'Terminal depuración'.



“Hola Mundo”: al servicio de depuración

- Modifica main.c para incorporar el siguiente código
 - (el printf() sale por un servicio de depuración)

```
#include <stdio.h>

void retardo(uint32_t cuenta);

int main(void)
{
    uint32_t i = 0;

    while (1)
    {

        retardo(20000000);
        printf("Contador %d\n", (int)i);
        i++;
    }
}

void retardo(uint32_t cuenta)
{
    while(cuenta--) {};
}
```



