Sistemas Electrónicos Digitales

Tema #3

1. Introducción

Temario

1. Introducción
2. GPIO: General Purpose Input/Output
3. Arquitectura Arm Cortex-M4
4. Interrupciones
5. C en ensamblador
6. Temporizadores (Timers)
7. Direct Memory Access
8. Comunicaciones Serie
9. Conversores A/D y D/A

Introducción

INTRODUCCIÓN A SISTEMAS EMBEBIDOS Y ARM

Sistemas embebidos

- ¿Qué es un sistema embebido?
  - **Es un sistema basado en ordenador orientado a una aplicación específica. Por tanto podemos decir que es un ordenador diseñado para un cometido concreto**.
  - Suele ser un componente más dentro de un sistema.

- ¿Por qué incluir un sistema embebido en un sistema mayor?
  - Mayores prestaciones y funcionalidad.
  - Mejor desempeño.
  - Menor coste.

Opciones para el diseño de sistemas embebidos

	Implementación	Coste del Diseño	Coste Unitario	Upgrades & Bug Fixes	Tamaño	Peso	Consumo	Velocidad del sistema
Hardware Dedicado	Lógica Discreta	BAJO	MEDIO	DIFICIL	GRANDE	ALTO	?	MUY RAPIDO
	ASIC	ALTO ($500K/ mask set)	MUY BAJO	DIFICIL	ENANO – 1 CHIP	MUY BAJO	BAJO	EXTREMA-DAMENTE RAPIDO
	Lógica Programable – FPGA, PLD	BAJO	MEDIO	FACIL	PEQUEÑO	BAJO	MEDIO A ALTO	MUY RAPIDO
Hardware genérico programable	Sistema basado en Microprocesador	BAJO A MEDIO	MEDIO	FACIL	PEQUEÑO A MEDIANO	BAJO A MODE RADO	MEDIO	MODERADO
	Microcontrolador	BAJO	MEDIO A BAJO	FACIL	PEQUEÑO	BAJO	MEDIO	LENTO A MODERADO
	PC embebido	BAJO	ALTO	FACIL	MEDIO	MODE RADO A ALTO	MEDIO A ALTO	RAPIDO

Microcontrolador vs. Microprocesador

- Ambos tienen una CPU que ejecuta instrucciones
- Los microcontroladores tienen periféricos que permiten realizar funciones de interfaz y control de forma concurrente (paralelo)
  - Conversión Analógica
  - Señales digitales
  - Temporización
  - Generadores de reloj
  - Comunicaciones
  - Fiabilidad y seguridad

Ejemplo de sistema embebido

Asistente para bicicleta

- Funciones
  - Medida de velocidad y distancia
- Entradas
  - Tacómetro
  - Teclado propio
- Restriciones
  - Tamaño
  - Coste
  - Consumo
  - Peso
- Salidas
  - Pantalla

Posible solución: microcontrolador de bajo coste (8 bits, 10MIPS)

Controlador de motor de un vehículo

- Funciones
  - Control del motor
  - Comunicación con el resto de sistemas del coche
  - Monitorización de la corriente
  - Detección de la velocidad de rotación.
- Restriciones
  - Fiabilidad en entorno agresivo
  - Coste
  - Peso
- Entradas y Salidas
  - Múltiples sensores y actuadores

Posible solución: microcontrolador de altas prestaciones (32 bits, 256 KB de memoria flash, 80 MHz)

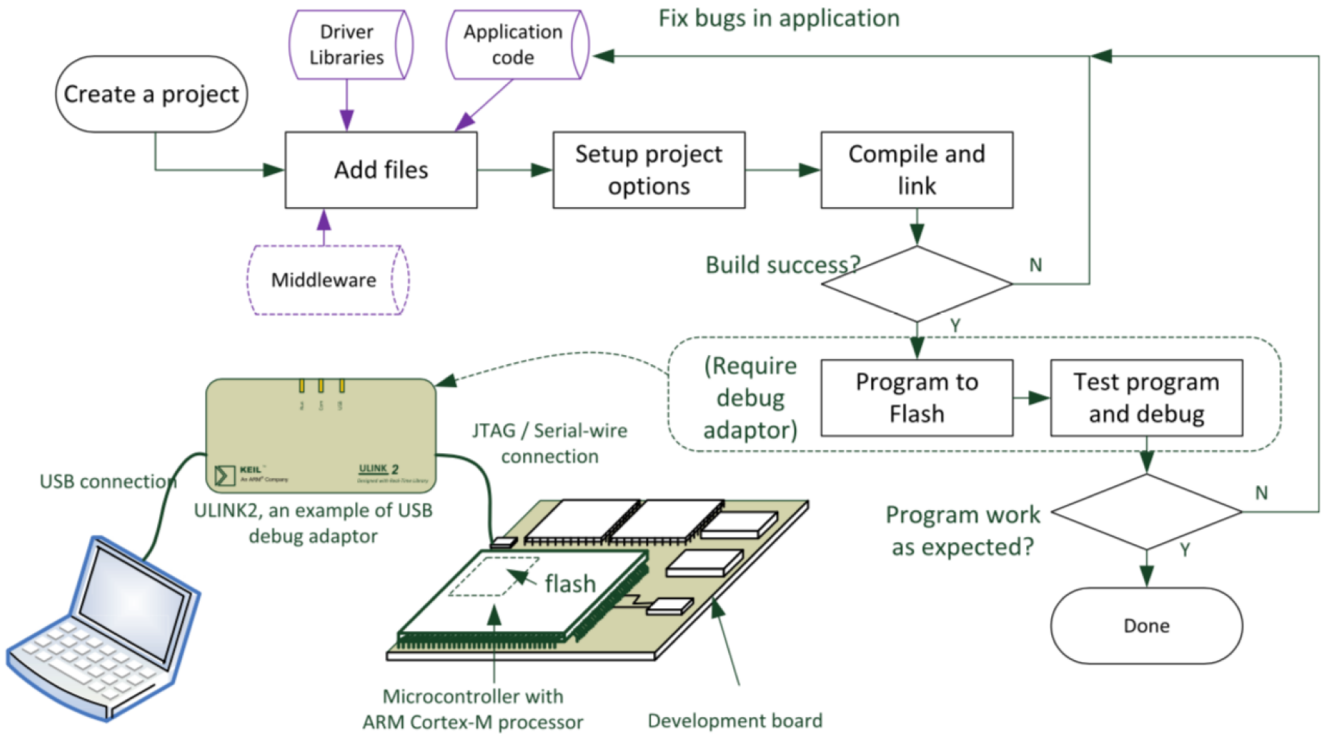
¿ Por qué ...

- Se usa C en vez de Java (o Python, o cualquier otro lenguaje de programación)?
  - C es el standard para sistemas embebidos porque:
    - Permite un control preciso de lo que hace el procesador.
    - Presenta requerimientos modestos de ROM, RAM, y MIPS (sistema más barato)
    - Comportamiento predecible, sin S.O.
- Aprender lenguaje ensamblador?
  - El compilador traduce las sentencias de C a ensamblador.
  - Para saber si el compilador traduce eficientemente, es necesario comprender aquello que genera.
  - A veces es necesario mejorar la ejecución escribiendo la parte crítica del código en ensamblador.
- Vais a necesitar un sistema de desarrollo?
  - Se aprende haciendo más que oyendo.

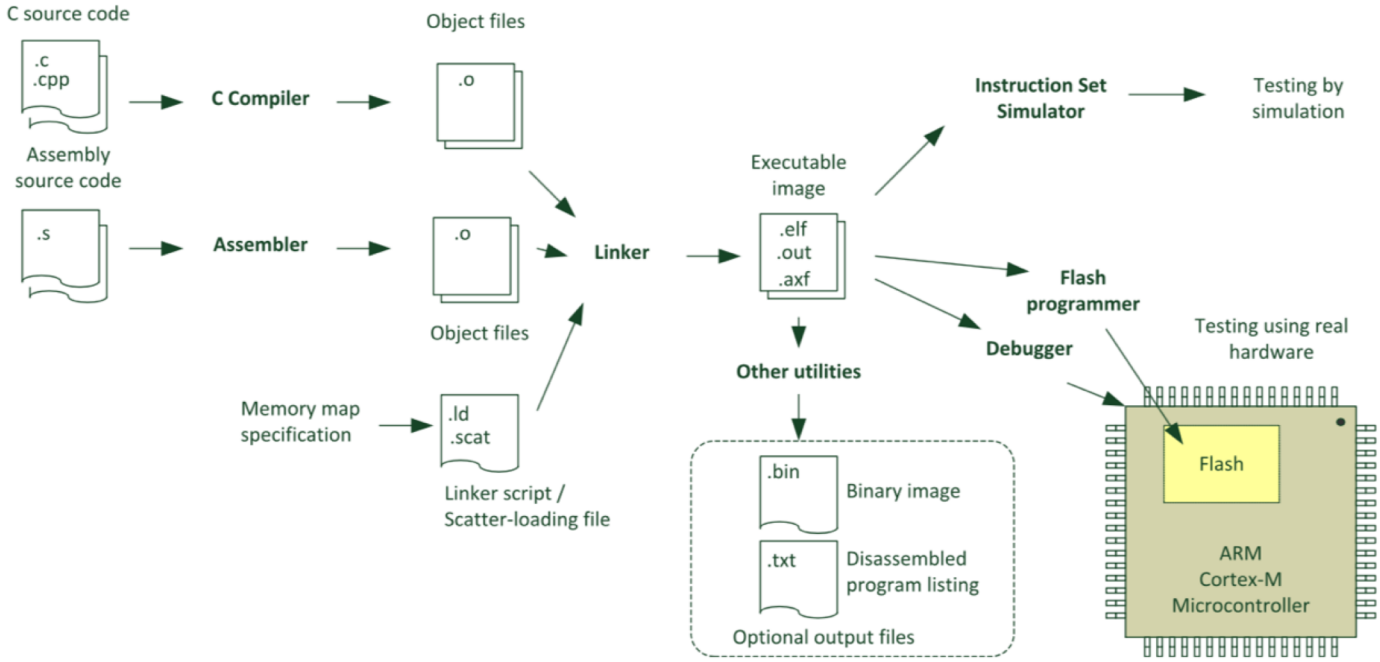
Sobre ARM

- ARM no fabrica procesadores ni vende los chips directamente.
- ARM licencia los diseños de los procesadores a los socios comerciales
- Basado en los diseños de procesadores ARM estos socios crean sus procesadores, microcontroladores y soluciones de sistema en chip.
- Este modelo de negocio se denomina comúnmente licencia de IP.
- Además de los diseños de procesador, ARM también licencia IP a nivel de sistema, como periféricos y controladores de memoria

Flujo de desarrollo software



Flujo de desarrollo software II



Introducción

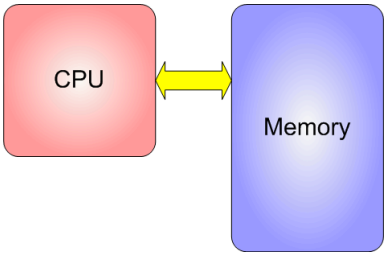
CONCEPTOS BÁSICOS

Arquitecturas básicas

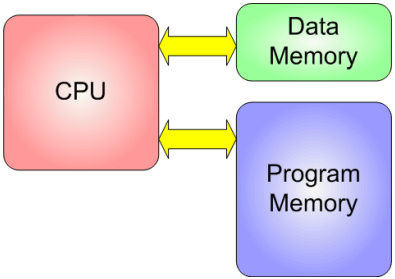
- Se identifican dos grandes tipos de arquitecturas en la realización de los microprocesadores dependiendo de los diferentes buses y acceso a éstos tanto de forma interna como externa con que cuenta un microprocesador:
  - **Arquitectura de Von Neumann**
    - También llamada Princeton
  - **Arquitectura Harvard**

Arquitecturas básicas II

- **Arquitectura de Von Neumann**
  - puede leer una instrucción o leer/escribir datos desde/hacia la memoria pero no al mismo tiempo
  - las instrucciones y datos usan el mismo sistema de buses



- **Arquitectura Harvard**
  - puede tanto leer una instrucción como realizar un acceso a la memoria de datos al mismo tiempo
  - cuenta con buses de acceso a programa y datos separados



Arquitecturas básicas III

Harvard

- Dos memorias con dos buses permiten el acceso paralelo a los datos y las instrucciones
- La unidad de control para dos buses es más complicada y más costosa
- El programa no puede escribirse solo.
- Ambas memorias pueden tener diferentes tamaños
- El desarrollo de una Unidad de Control complicada necesita más tiempo.
- La memoria de datos y la de instrucciones no se pueden compartir

Von Neumann

- Si el contenido de la memoria está organizado toda la memoria instalada se puede usar.
- Un solo bus simplifica el diseño de la unidad de control.
- La computadora con un bus es más barata.
- Un error en un programa puede sobrescribir instrucciones y bloquear el programa.
- Se accede a los datos y a la instrucción de la misma manera.
- Un mismo bus para datos, instrucciones y dispositivos, es un cuello de botella.

Arquitecturas básicas IV

- ✓ **Harvard** se utiliza principalmente para sistemas embebidos y procesamiento de señales (DSP).

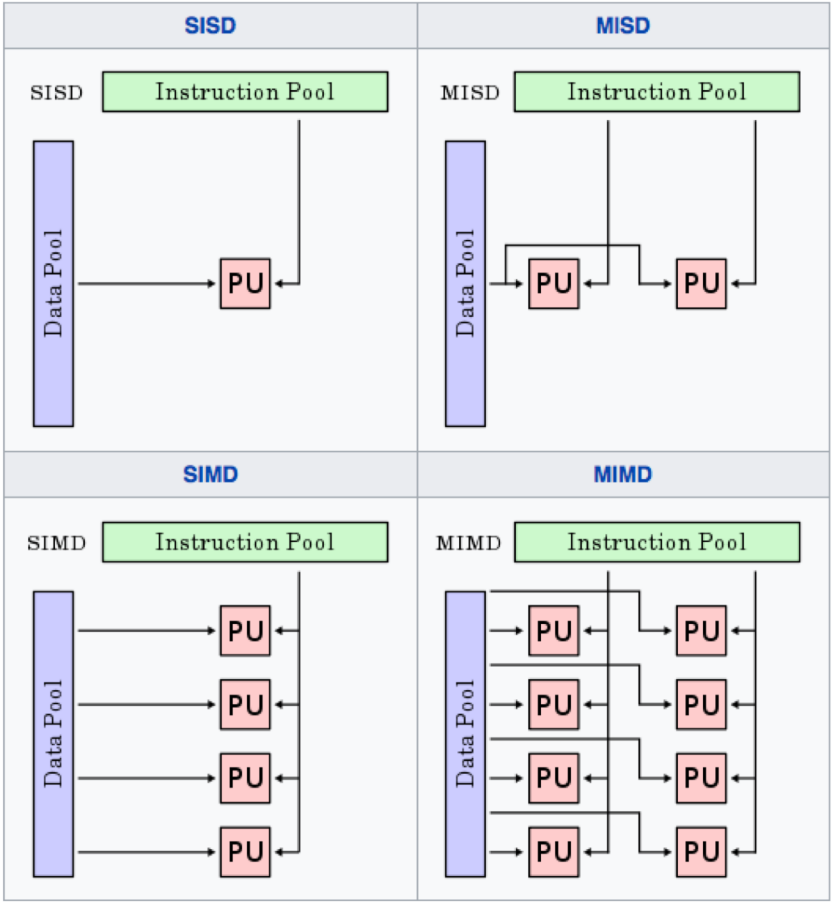
- ✓ **Von Neumann** es mejor para computadoras de escritorio, computadoras portátiles, estaciones de trabajo y computadoras de alto rendimiento.

Taxonomía de Flynn

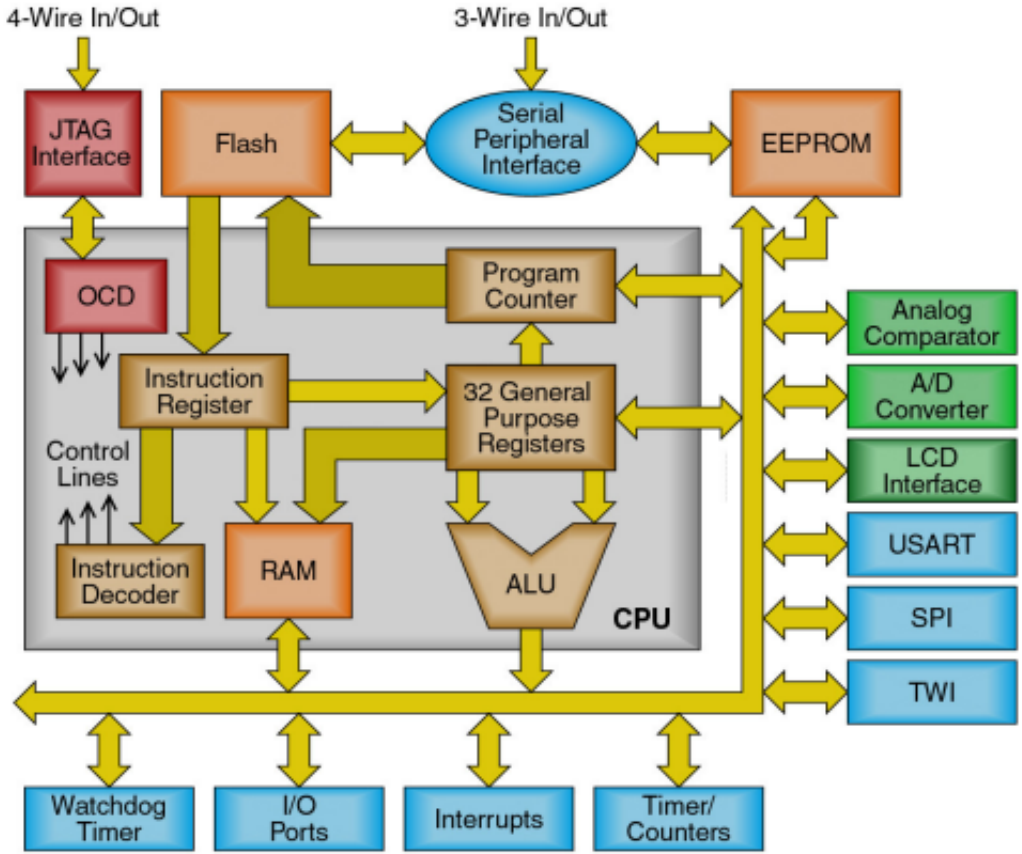
- La taxonomía de Flynn es una clasificación de arquitecturas de computadores propuesta por Michael J. Flynn en 1972
  - **SISD**
    - Single Instruction, Single Data stream
  - **SIMD**
    - Single Instruction, Multiple Data streams
  - **MISD**
    - Multiple Instruction, Single Data stream
  - **MIMD**
    - Multiple Instruction, Multiple Data streams

Taxonomía de Flynn

ARM CORTEX M4



Estructura básica de un Microcontrolador



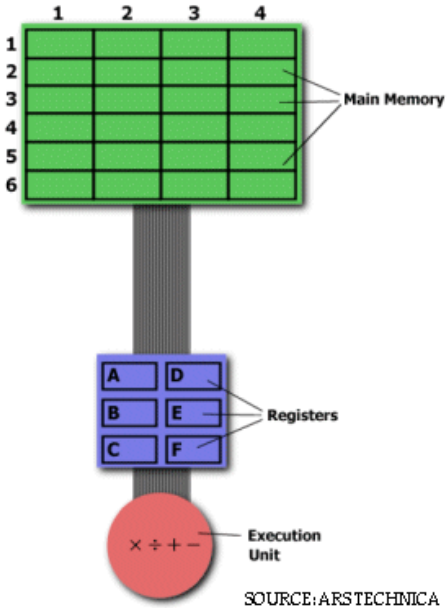
RISC vs CISC

- CISC: Complex Instruction Set Computer

```
MULT 2:3, 5:2
```

- RISC: Reduced Instruction Set Computer

```
LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, A
```



Microcontroladores 8 bits, 16 bits, 32 bits

- El tamaño de la **palabra** es un aspecto importante en la arquitectura de procesadores.
- La mayoría de los **registros** de un Microprocesador / Microcontrolador tienen el tamaño de la palabra
- Las **operaciones** que hace la ALU es manejando **operandos** cuyo tamaño es el tamaño de la palabra, así como la cantidad de **datos transferidos** a memoria y dirección utilizada para designar una localización de memoria a menudo ocupa una palabra.
- También los valores que pueden tomar las **variables** dependen del tamaño de la palabra.

Memoria Flash, SRAM y EEPROM

- **SRAM**
  - **Variables locales, datos parciales**. Usualmente se trata como banco de registros. Memoria volátil. Es un recurso limitado y debemos supervisar su uso para evitar agotarlo.
- **Flash**:
  - **Memoria de programa**. Usualmente desde 1 Kb a 4 Mb (controladores de familias grandes).
- **EEPROM**:
  - Se puede grabar desde el programa del microcontrolador. Usualmente, **constantes de programa**. Memoria no volátil para mantener datos después de un reset.
  - Las EEPROMs tienen un número limitado de lecturas/escrituras, tener en cuenta a la hora de usarla.

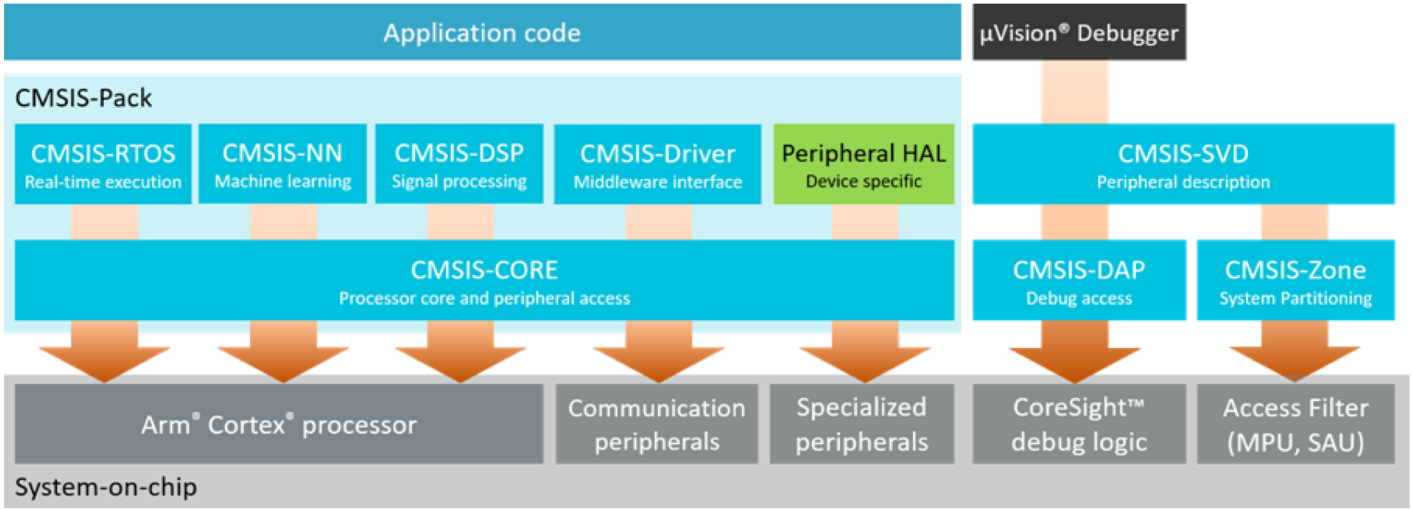
Introducción

PROGRAMANDO NUESTRO MICRO

Programando el Micro

- Podemos utilizar básicamente dos lenguajes de programación de micros:
  - **Ensamblador**
  - **C**
- A la hora de programar un micro podemos tomar dos caminos:
  - Acceder a los registros **directamente en la memoria**
  - Utilizar herramientas (**librerías**) de alto nivel
    - En el caso de ARM y ST: CMSIS y HAL (y mbed)

CMSIS - HAL



- CMSIS: Cortex Microcontroller Software Interface Standard
- HAL: Hardware Abstraction Layer

CMSIS

- CMSIS (**Cortex Microcontroller Software Interface Standard**) proporciona un **marco de trabajo software** para aplicaciones embebidas que se ejecutan en microcontroladores basados en Cortex-M y A5 / A7 / A9.
- CMSIS se inició en 2008 en una cooperación con varios proveedores de software y silicio.
- CMSIS permite interfaces de software consistentes y simples para el procesador y los periféricos, lo que simplifica la **reutilización** del software, reduciendo la **curva de aprendizaje**

HAL

- HAL (**Hardware Abstraction Layer**) es un driver que proporciona un conjunto simple genérico de **APIs** para interactuar con la capa superior (aplicaciones, bibliotecas y pilas).
- Permite que las capas incorporadas (**middleware**) implementen sus funciones sin conocer en profundidad cómo usar la MCU.
- Esta estructura mejora la **reutilización** del código de la biblioteca y garantiza una **fácil portabilidad** a otros dispositivos.

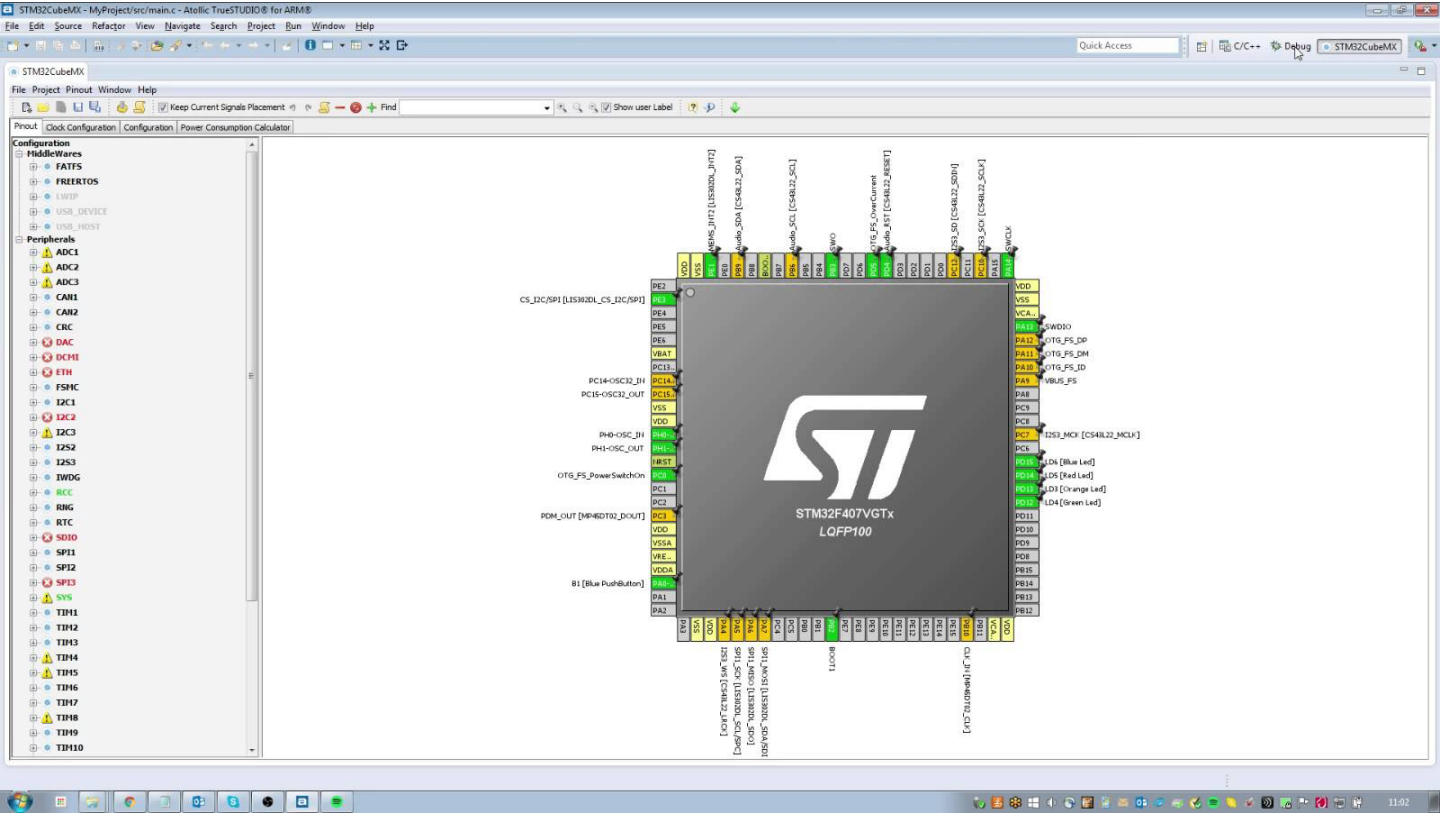
STM32Cube

- A software configuration tool that allows generating C initialization code (also using graphical wizards).
- A comprehensive embedded software platform, delivered per series (such as STM32CubeF4 for STM32F4 Series),including:
  - The STM32Cube hardware abstraction layer (HAL), an STM32 abstraction layer embedded software, ensuring maximized portability across the STM32 portfolio
  - The Low Layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. The LL APIs are available only for a set of peripherals
  - A consistent set of middleware components such as RTOS, USB, TCP/IP, Graphics

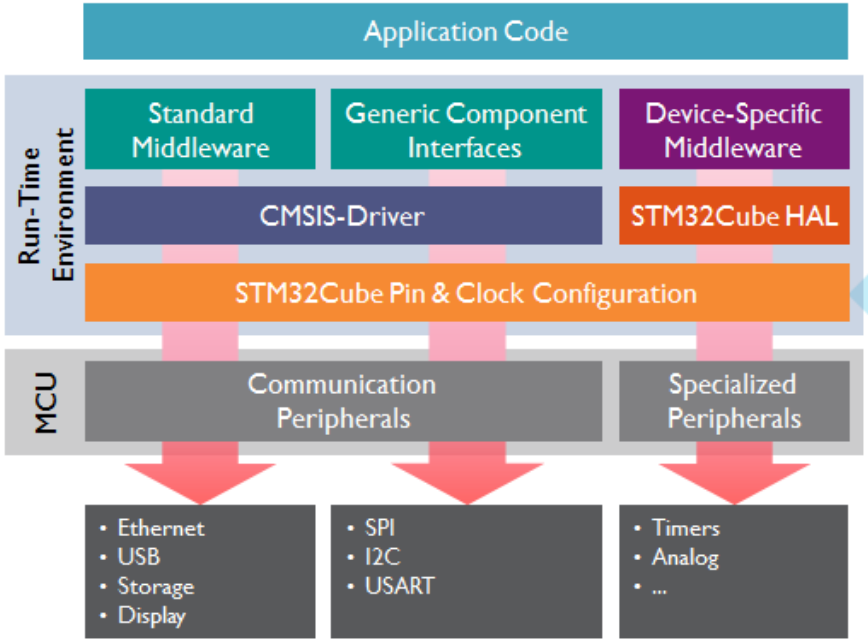
STM32Cube

- Una herramienta de configuración de software que permite generar código de inicialización C (también usando asistentes gráficos).
- Una plataforma integral de software embebido, por series (como STM32CubeF4 para la serie STM32F4), que incluye:
  - La capa de abstracción de hardware STM32Cube (HAL)
  - APIs de capa baja (LL): rápidas, ligeras y orientadas a expertos, que estás más cerca del hardware que HAL (disponibles solo para un conjunto de periféricos)
  - Un conjunto consistente de componentes de middleware como RTOS, USB, TCP / IP, gráficos

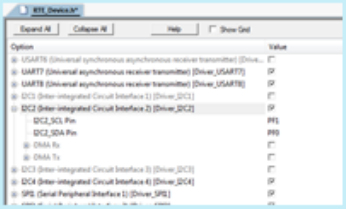
STM32CubeMX



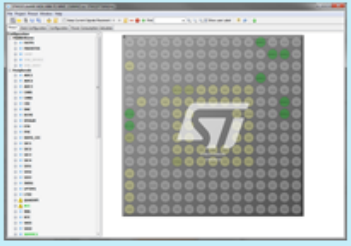
STM32CubeMX



Option 1: RTE_Device.h

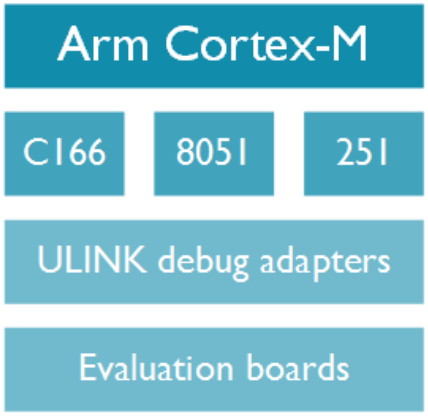


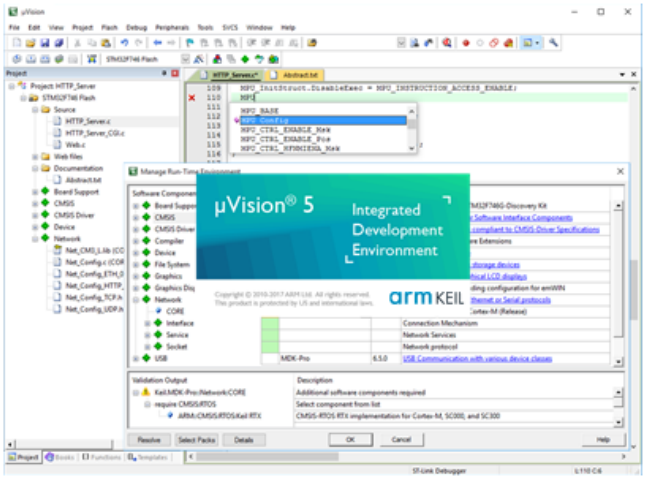
Option 2: STM32CubeMX



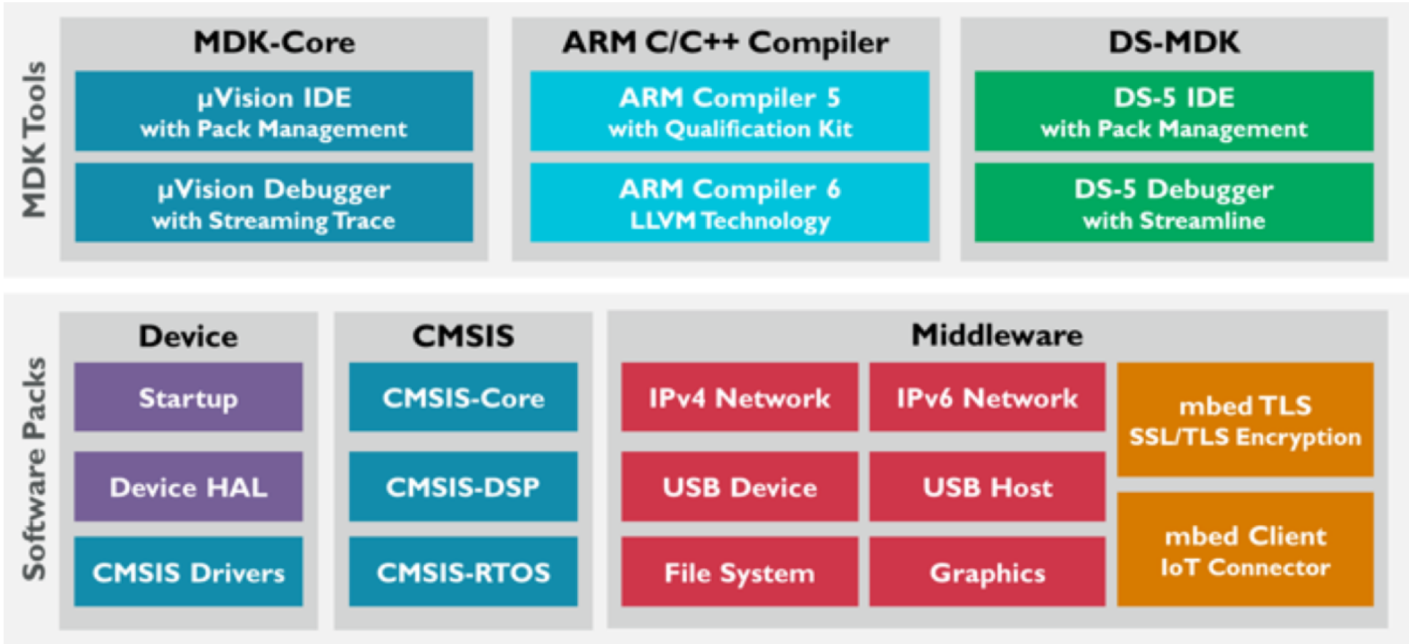
ARM Keil

- Keil® MDK es una solución de desarrollo de software para microcontroladores basados en Arm® e incluye componentes para crear, construir y depurar aplicaciones embebidas.





ARM Keil


MDK Tools
MDK-Core
µVision IDE
with Pack Management
µVision Debugger
with Streaming Trace
ARM C/C++ Compiler
ARM Compiler 5
with Qualification Kit
ARM Compiler 6
LLVM Technology
DS-MDK
DS-5 IDE
with Pack Management
DS-5 Debugger
with Streamline
Software Packs
Device
Startup
Device HAL
CMSIS Drivers
CMSIS
CMSIS-Core
CMSIS-DSP
CMSIS-RTOS
Middleware
IPv4 Network
IPv6 Network
USB Device
USB Host
File System
Graphics
mbed TLS
SSL/TLS Encryption
mbed Client
IoT Connector

Fin