

MICROS Y LABORATORIO INTRODUCCION

PROFESOR: ROBINSON JIMENEZ MORENO



NOMBRE DE LA ASIGNATURA	MICROS Y LABORATORIO
CÓDIGO	18203
SEMESTRE	V

OBJETIVO GENERAL

Diseñar e implementar sistemas embebidos basados en Microcontroladores, integrando herramientas de hardware y software, orientando su aplicación a sistemas mecatrónicos.

COMPETENCIA GLOBAL

- *El estudiante generará habilidad en el análisis de los sistemas Microprocesados, Microcontrolados y sistemas embebidos basados en estos.*
- *Podrá seleccionar la tecnología de procesador más adecuada para aplicar en los diferentes procesos de control industrial.*
- *Desarrollará habilidades en la programación de un Sistema embebidos basado en microcontrolador*



CONTENIDO

Semana	Tema o actividad presencial	Actividades de trabajo independiente
1	Introducción al curso, metodología de trabajo. Introducción a la arquitectura de microcontroladores / Lenguaje C orientado a Microcontroladores	Estudiar e identificar las diferentes arquitecturas microcontroladas. Realizar conversiones entre sistemas de codificación numérica binario-hexa y decimal. Repasar uso de sentencias condicionales en C.
2	Puertos de entrada/salida. Modulo GPIO. Ejemplos y aplicaciones GPIO	Identificar los registros de propósito general para configuración de entradas salidas digitales. Desarrollar aplicaciones básicas microcontroladas.
3	Interrupciones Ejemplo y aplicaciones.	Resolver problemas de control microcontrolado basado en interrupciones.
4	Aplicaciones: Visualización dinámica, Display n-segmentos, Matrices de LEDs. Aplicaciones: Teclado Matricial (Interrupción)	Desarrollar programas de manejo de dispositivos de entrada y salida digital.
5	Display LCD – alfanumérico. Display LCD – gráfico.	Desarrollar programas de manejo de dispositivos de visualización digital, integrar interfaces de usuario.
6	Ejercicios de refuerzo/ Taller pre-parcial Primer Parcial	Replicar los programas vistos en clase bajo la solución de problemas.
7	Modulo USART (Comunicación Serial) Registros y contadores. Taller: Aplicaciones UART	Diseñar sistemas básicos de transferencia de información serial.
8	Manejo de motores paso a paso. Manejo de servomotores. Aplicaciones serial – inalámbrico.	Desarrollar aplicaciones microcontroladas basadas en el uso de motores.
9	Modulo ADC Taller: Aplicaciones modulo ADC	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos ADC.

10	Modulo DAC Taller: Aplicaciones modulo DAC	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos DAC.
11	Modulo TIMER: PWM Taller: Aplicaciones Timer PWM	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos TIMER - PWM.
12	Ejercicios de refuerzo/ Taller pre-parcial Segundo Parcial	
13	Modulo TIMER: OC (Output Capture) Taller: Aplicaciones Timer (OC)	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos TIMER-OC.
14	PLL Taller: Aplicaciones PLL	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos PLL.
15	Modulo I2C Taller: Aplicaciones I2C	Desarrollar aplicaciones microcontroladas basadas en el uso de módulos I2C.
16	Ejemplos de aplicación Taller integrador	

SISTEMA DE EVALUACIÓN

Derivado de cada uno de los cortes de notas establecidos en el calendario académico se evaluarán los diferentes temas según el avance del contenido programático mediante las siguientes herramientas:

- Parciales teórico prácticos.
- Talleres y quices en clase.
- Presentación de laboratorios.

Las evaluaciones teórico-prácticas equivalen al 100% de la nota total, se realizan por medio de dos notas cada uno correspondiente al 21%, de la nota final del 28% y la nota de las prácticas desarrollados en el laboratorio del 30 %.

BIBLIOGRAFÍA

1. Índice con referencias de páginas y citas bibliográficas
2. Libros textos
 - *Tammy Noergaard, Embedded Systems Architecture, Second Edition: A Comprehensive Guide for Engineers and Programmers, 2012.
 - *Jason D. Bakos Embedded Systems: ARM Programming and Optimization, 2015
 - *Título: Manual de usuario de la tarjeta STM32F407
(datasheet <http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf>)
3. Libros electrónicos
<http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1577/LN11>



EVALUACIÓN DEL CURSO

3 CORTES	CONCEPTO	VALOR (por corte)
	EXAMEN ESCRITO	50%
	TALLERES	25%
	TAREAS Y QUICES	25%

El examen escrito tiene carácter individual.

Los talleres tienen carácter grupal.

Las tareas tienen carácter individual, las tareas se solicitarán únicamente al inicio de la clase de forma aleatoria, la no presentación de una tarea por no encontrarse en el salón solo se justificara con excusa escrita validada por la dirección de mecatrónica.

Contacto: robinson.jimenez@unimilitar.edu.co

Se llamará a lista una única vez al inicio de cada clase, para registro de fallas.



Atención a estudiantes

Asistente Investigación: Javier Orlando Pinzón Arenas

Ing. Mecatrónico, Magister en Ingeniería mecatrónica.

Miércoles 2 a 4 pm Programa de ingeniería mecatrónica.

Monitora Micros:

Anguie Yomara García Burgos

Estudiante sexto semestre Ingeniería mecatrónica.

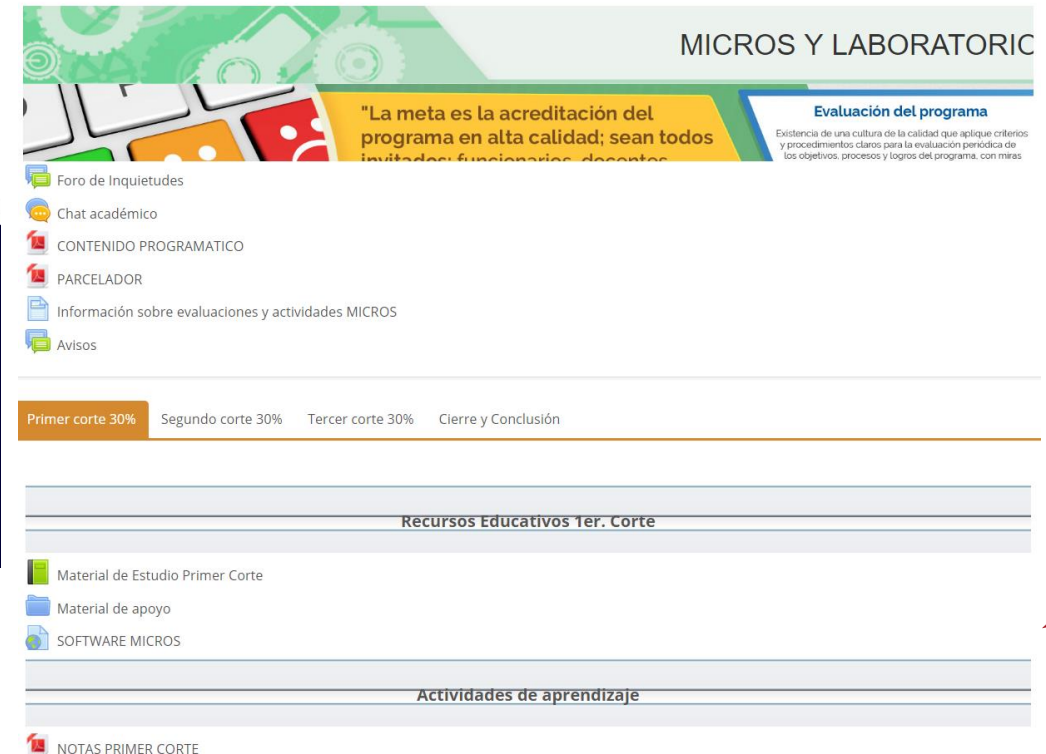
Viernes 2 a 4 pm Programa de ingeniería mecatrónica.

Material del curso



The screenshot shows the Moodle course interface. At the top, the URL is <https://virtual2.umng.edu.co/moodle/course/index.php?categoryid=246>. The page header includes the 'Aulas Virtuales' logo and the user name 'ROBINSON JIMENEZ MORENO'. Below the header, there is a navigation bar with 'Inicio', 'Cursos', and 'Espacio personal'. The main content area shows the course title 'MICROS Y LABORATORIO' and a list of activities: 'Foro de inquietudes', 'Chat académico', 'CONTENIDO PROGRAMATICO', 'PARCELADOR', 'Información sobre evaluaciones y actividades MICROS', and 'Avisos'. The course progress bar shows 'Primer corte 30%', 'Segundo corte 30%', 'Tercer corte 30%', and 'Cierre y Conclusión'. The 'Recursos Educativos 1er. Corte' section includes 'Material de Estudio Primer Corte', 'Material de apoyo', and 'SOFTWARE MICROS'. The 'Actividades de aprendizaje' section is also visible.

MICROS Y LAB



The screenshot shows the course content page for 'MICROS Y LABORATORIO'. The page features a banner with the text 'La meta es la acreditación del programa en alta calidad; sean todos invitados: funcionarios, docentes, estudiantes, padres de familia'. Below the banner, there is a list of activities: 'Foro de inquietudes', 'Chat académico', 'CONTENIDO PROGRAMATICO', 'PARCELADOR', 'Información sobre evaluaciones y actividades MICROS', and 'Avisos'. The course progress bar shows 'Primer corte 30%', 'Segundo corte 30%', 'Tercer corte 30%', and 'Cierre y Conclusión'. The 'Recursos Educativos 1er. Corte' section includes 'Material de Estudio Primer Corte', 'Material de apoyo', and 'SOFTWARE MICROS'. The 'Actividades de aprendizaje' section is also visible.

Fechas importantes:

Primer parcial 4 de septiembre

Segundo parcial 16 de octubre

Examen final 27 de noviembre



Sistemas embebidos microcontrolados

- Un sistema embebido es un sistema electrónico diseñado para realizar funciones específicas en tiempo real. Al tratarse de un dispositivo microcontrolado, significa que posee un circuito integrado que se encarga de controlar los elementos de entrada y salida, además de un procesador y una memoria que permiten guardar el programa y sus variables



Lenguajes de programación

- Cada lenguaje de programación está diseñado para describir un conjunto de acciones que debe realizar un microcontrolador, entre los cuales se pueden encontrar los siguientes
 - Lenguaje de ensamblador (Assembler)
 - Lenguaje PIC-BASIC PRO (PBP)
 - Lenguaje C
 - Lenguaje Pascal

<https://prezi.com/aowwntefs3gy/lenguajes-de-programacion-para-microcontroladores/>



Estructura de programación en C

- Este lenguaje de programación permite programar de manera estructurada o no, y tener acceso a memoria de bajo nivel empleando punteros y diferentes tipos de datos como:

Nombre	Peso (Bytes)	Valor Inicial	Valor final
Signed char	1	-128	127
Unsigned char	1	0	255
Signed short	2	-32768	32728
Unsigned short	2	0	65535
Signed int	2	-2147483648	2147483647
Unsigned int	2	0	4294967295
Signed long	4	-32768	32768
Unsigned long	4	0	65535
Float	4	$3,4 \times 10^{-38}$	$3,4 \times 10^{38}$
Double	8	$1,7 \times 10^{-308}$	$1,7 \times 10^{-308}$
Long double	10	$3,4 \times 10^{-4932}$	$3,4 \times 10^{4932}$



Estructura de programación en C

- El lenguaje C tiene la ventaja de representar las operaciones aritméticas y lógicas con símbolos como:







Símbolo	Descripción
=	Asignación
+	Suma, adición
-	Resta, sustracción
*	Multiplicación, producto
/	Cociente división entera
%	Resto división entera
	División
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
==	Igual
!=	Diferente
&&	And, y, conjunción
	Or, o, disyunción
!	Not, no, negación
&	And bit a bit
	Or bit a bit
~	Complemento a uno o negación bit a bit
^	O-exclusiva bit a bit



Estructura de programación en C

➤ Diagramas de flujo:

- Los diagramas de flujo permiten representar de forma gráfica el proceso de un algoritmo de programación
- Usan símbolos para determinar los subprocesos y flechas para unirlos, de inicio a fin

Símbolo	Significado
	Inicio y fin del programa
	Representa la ejecución de una actividad o sub proceso
	Condición, Decisión o pregunta
	Enlace entre actividades o sub procesos por medio de otra actividad o procedimiento
	Guarda un documento o proceso de forma permanente
	Guarda un documento o proceso de forma temporal



Estructura de programación en C

➤ Sentencias condicionales:

- Son herramientas de programación que, basadas en una condición lógica, ejecutan una serie de instrucciones. Entre ellas se encuentran:
 - If
 - For
 - While
 - Do While
 - Switch Case



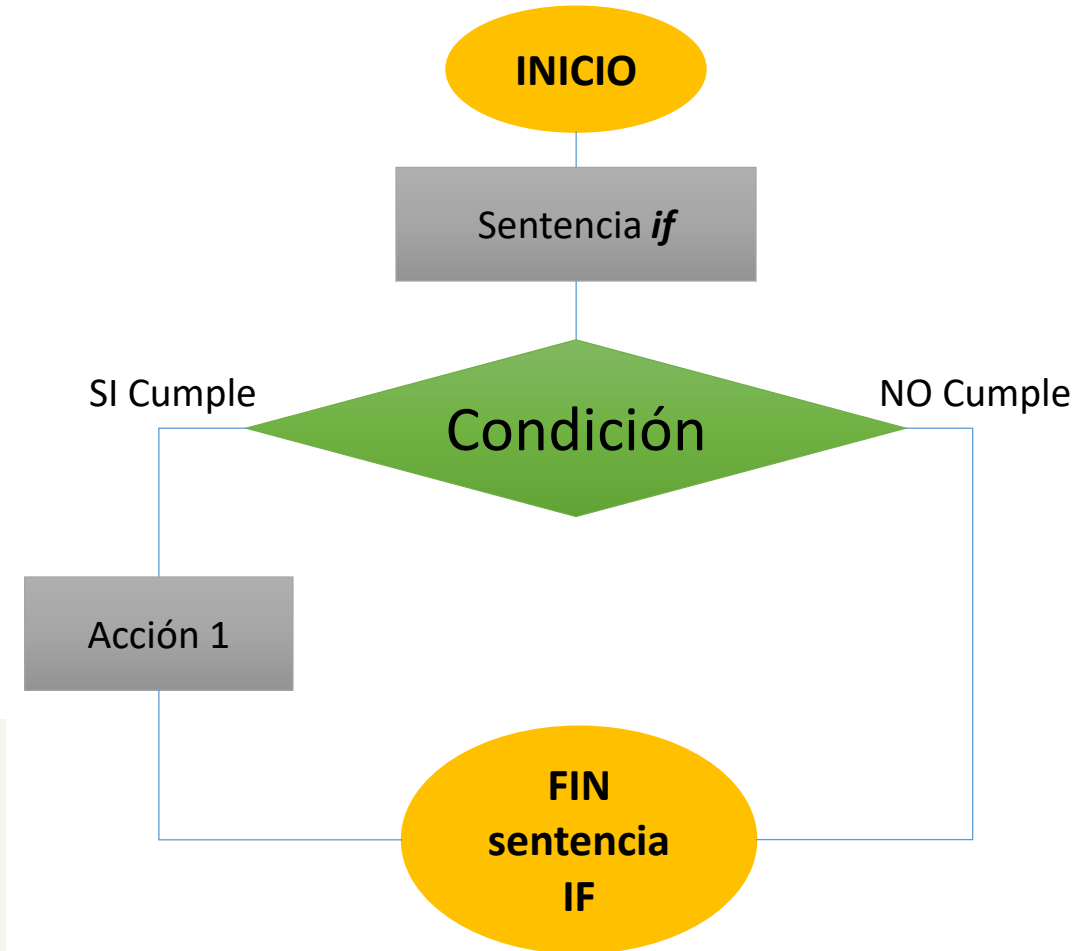
Estructura de programación en C



Estructura IF:

- Su función es evaluar una condición lógica donde, si la condición se cumple se ejecuta un código, en caso contrario, se ejecuta el código siguiente al IF.

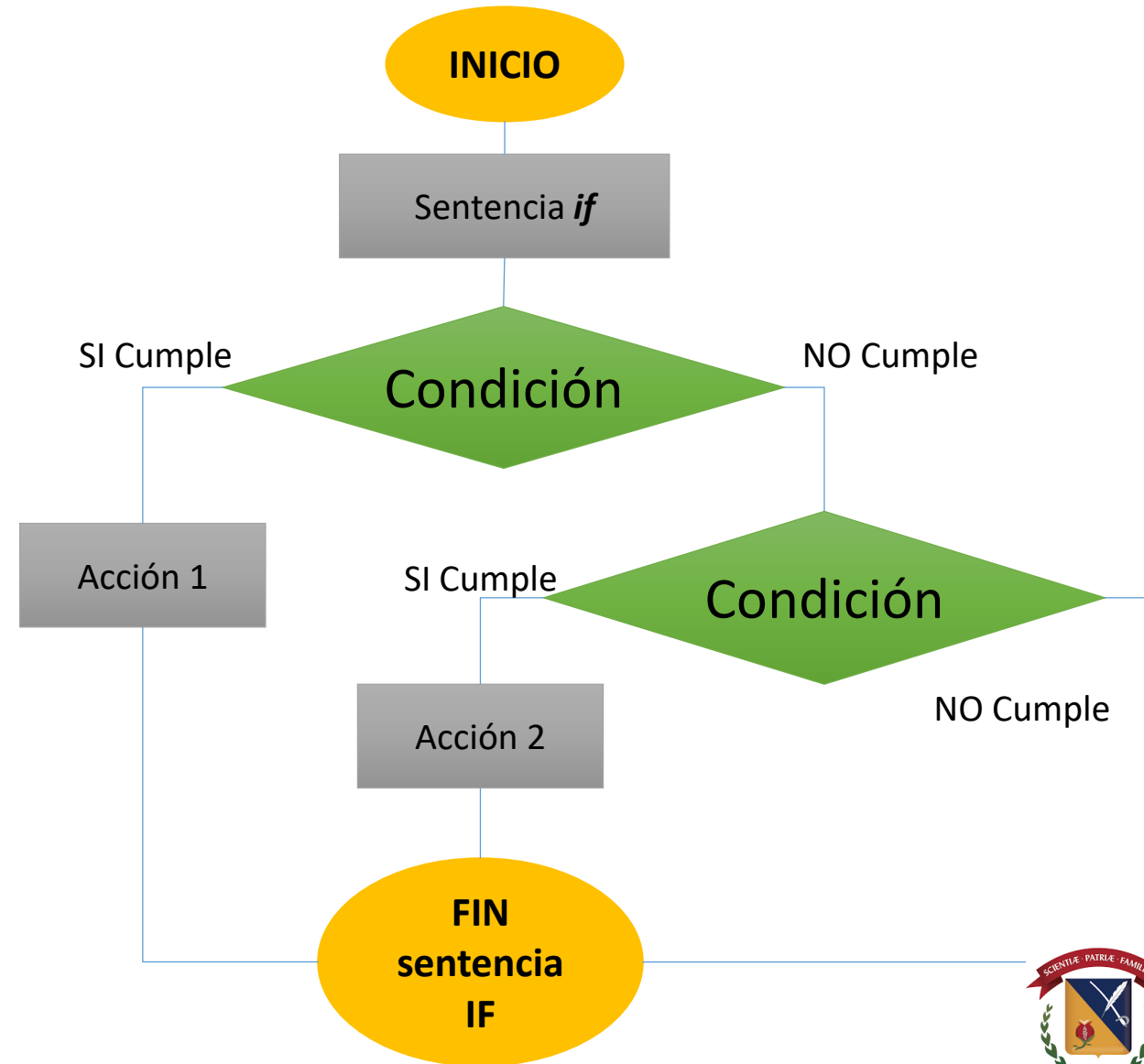
```
int valor;  
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10  
{  
    //Codigo que se ejecuta si se cumple la condición  
}
```



Estructura de programación en C

- **Estructura ELSE-IF:**
- Se utiliza para programar condiciones anidadas, dependientes o secundarias

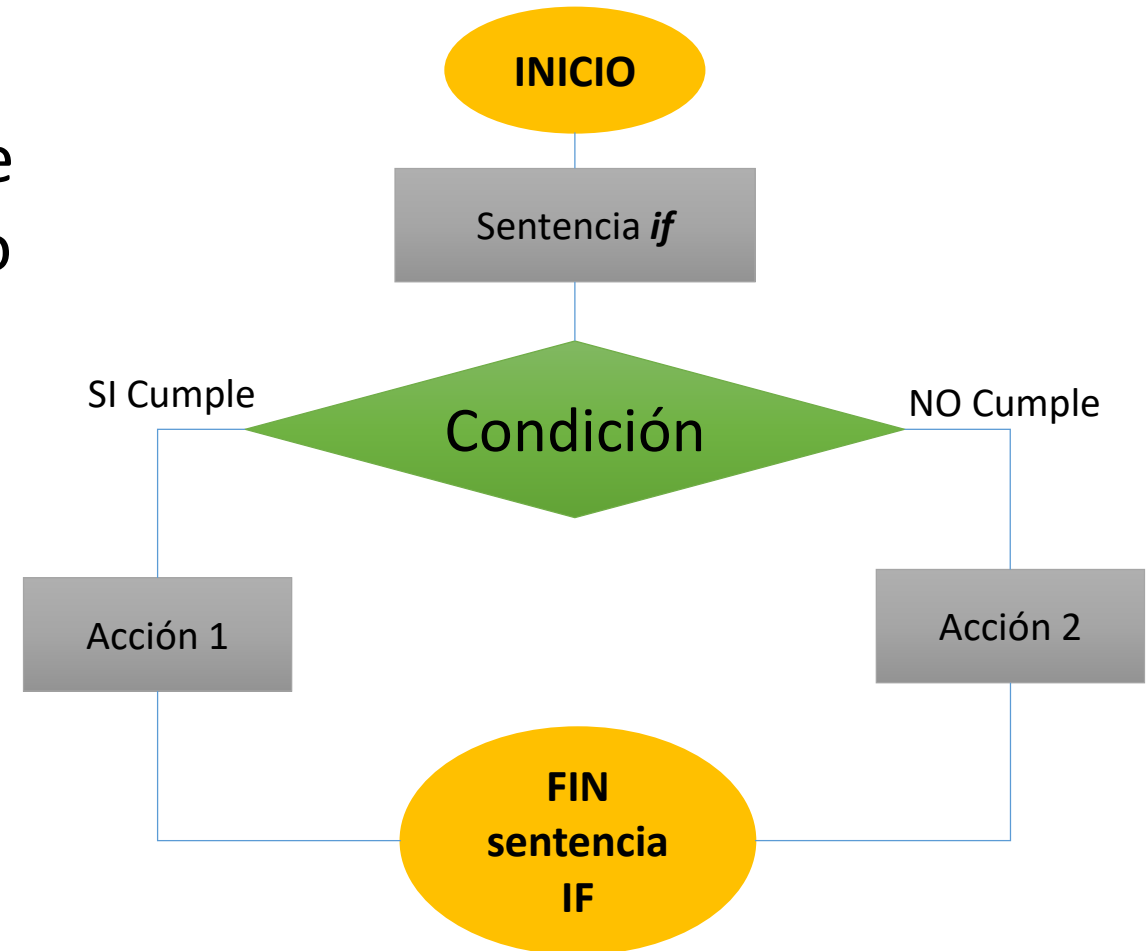
```
int valor;  
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10  
{  
    // Codigo que se ejecuta si se cumple esta condición  
}  
else if (valor == 20) // Condicion que se cumple si la variable valor es igual a 20  
{  
    // Codigo que se ejecuta si se cumple esta condición  
}  
else if (valor == 30) // Condicion que se cumple si la variable valor es igual a 30  
{  
    // Codigo que se ejecuta si se cumple esta condición  
}
```



Estructura de programación en C

- **Estructura ELSE-IF:**
- También se utiliza para casos en que se requiera ejecutar un código tanto cuando la condición es verdadera como cuando es falsa.

```
int valor;  
if(valor == 10) // Condicion que se cumple si la variable valor es igual a 10  
{  
    //Codigo que se ejecuta si se cumple esta condición  
}  
else  
{  
    //Codigo que se ejecuta si la condicion del if no se cumple  
}
```

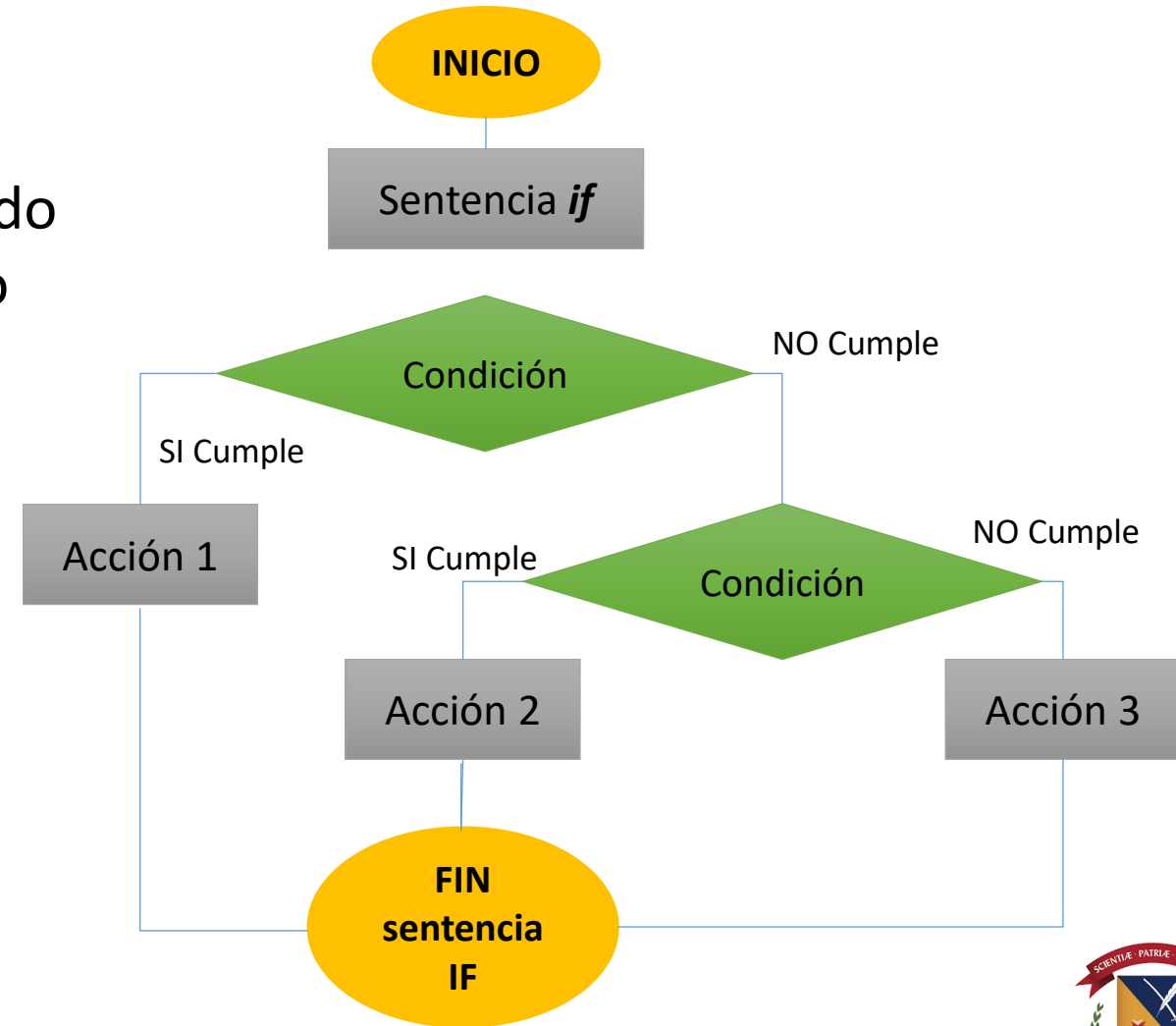


Estructura de programación en C

➤ Estructura ELSE-IF:

- También se utiliza para casos en que se requiera ejecutar un código tanto cuando la condición es verdadera como cuando es falsa.

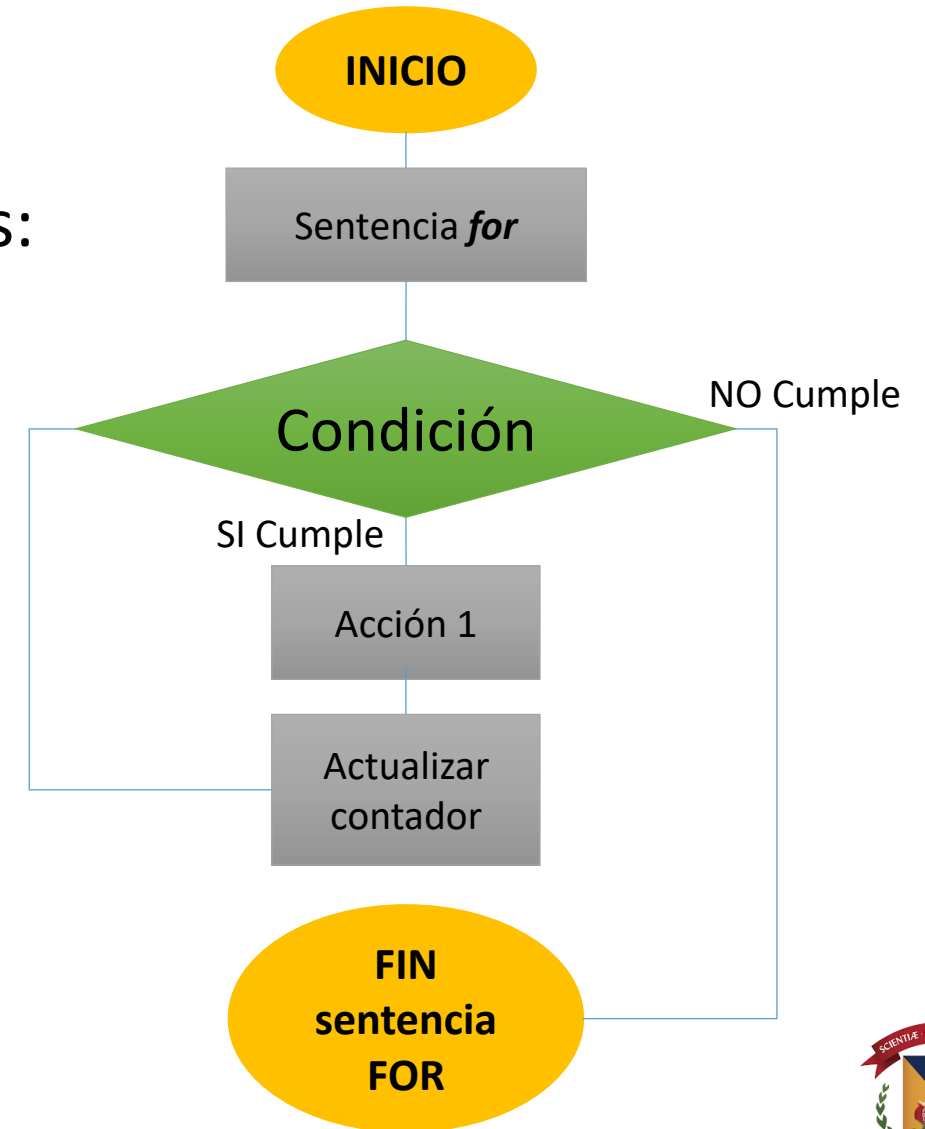
```
int valor;  
if(valor > 10) // Condicion que se cumple si la variable valor es mayor a 10  
{  
    //Codigo que se ejecuta si se cumple la condición del primer if  
    if(valor == 15) // Condicion que se cumple si la variable valor es igual a 15  
    {  
        //Codigo que se ejecuta si se cumple esta condición  
    }  
    else if (valor == 20) // Condicion que se cumple si la variable valor es igual a 20  
    {  
        //Codigo que se ejecuta si se cumple esta condición  
    }  
    else  
    {  
        //Codigo que se ejecuta si la condicion del segundo if y la del else if no se cumplen  
    }  
}
```



Estructura de programación en C

- **Estructura FOR:**
- Se utiliza para ejecutar una tarea iterativamente. Consta de tres parámetros:
 - Inicialización de variables
 - Condición que debe cumplir
 - Incremento o cambio de variables condicionales

```
//Declaracion de Variables
int numero1;
int numero2;
//Estructura del ciclo for
for( numero1=0, numero2=10; (numero1<10 && numero2>0) ; A++, B-- )
{
    //Codigo que se ejecuta cuando la condicion se cumple
}
```

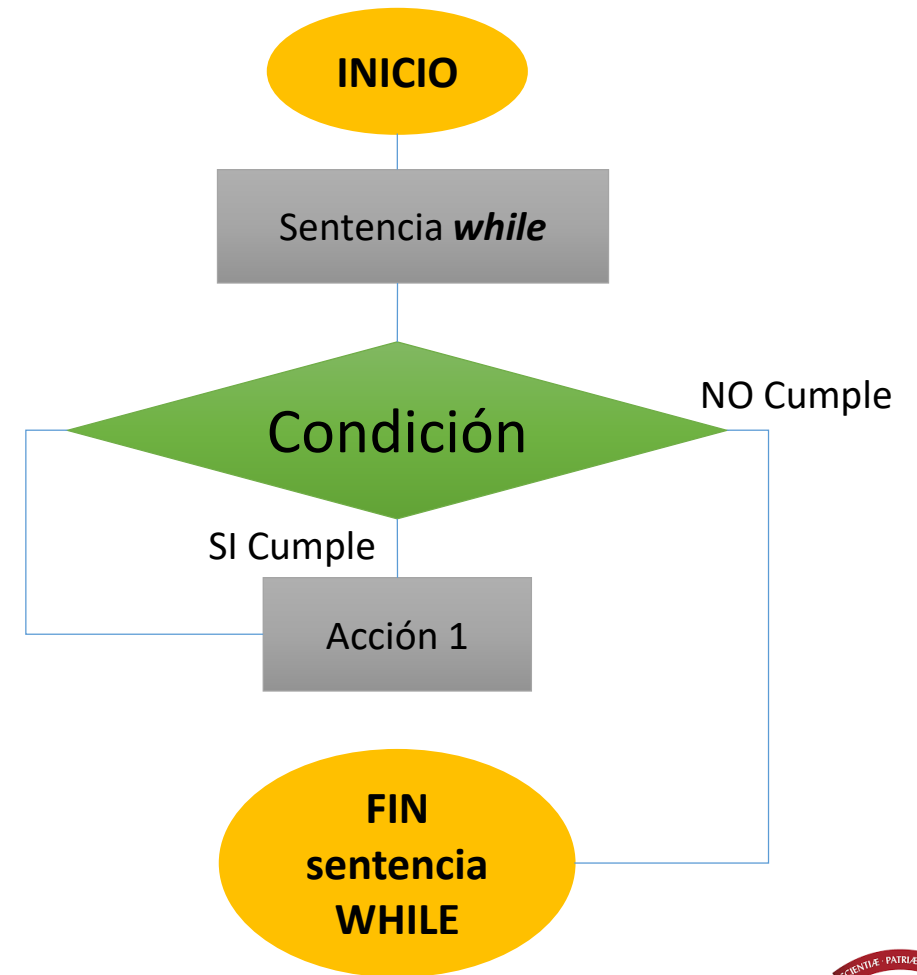


Estructura de programación en C

➤ Estructura WHILE y DO WHILE:

- Se utiliza para ejecutar una tarea iterativa de la cual no se conoce el número exacto de iteraciones requeridas antes de que la condición deje de ser verdadera

```
int valor=0; //Declaracion de la Variable
while(valor<10)// Declaracion del Ciclo que evaluara si valor es menor a 10
{
    valor++; // Mientras valor sea menor a 10, se incrementa en 1 esta variable
}
```

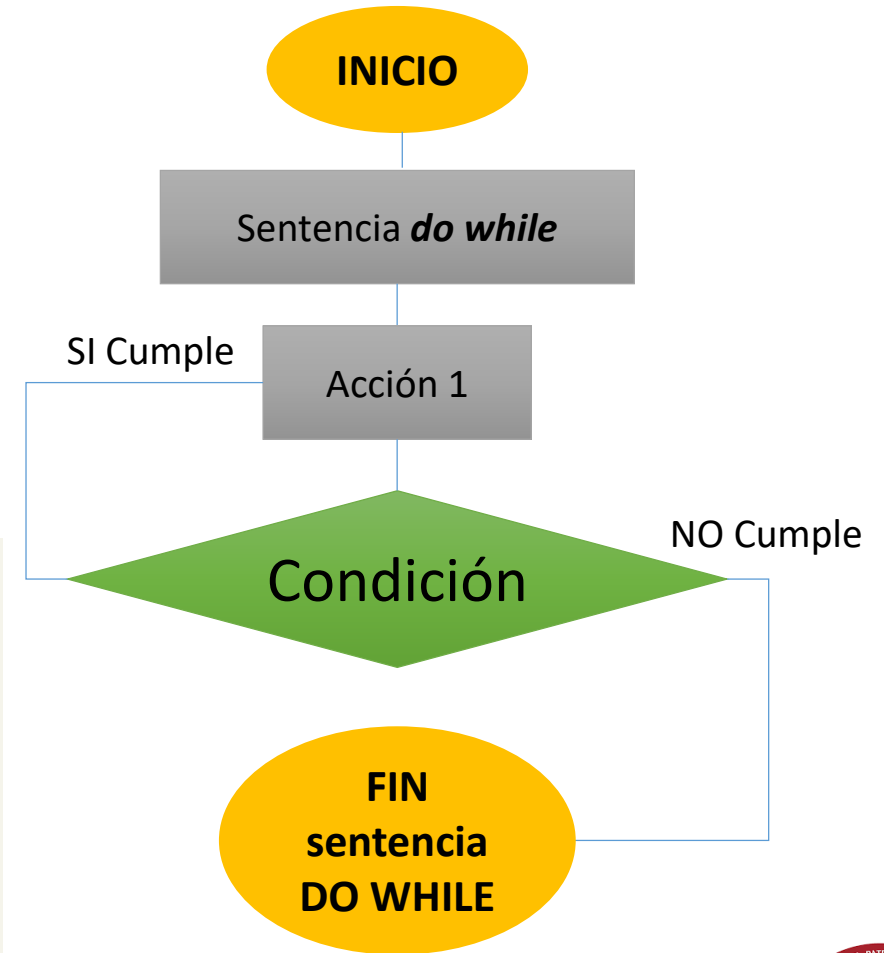


Estructura de programación en C

➤ Estructura WHILE y DO WHILE:

- En el caso del DO WHILE, se ejecuta primero el código y luego se evalúa si se cumple la condición.

```
int valor=0; //Declaracion de la Variable
do // Declaracion del Ciclo DO WHILE
{
    valor++; // Mientras valor sea menor a 10, se incrementa en 1 esta variable
}while(valor<10)
```

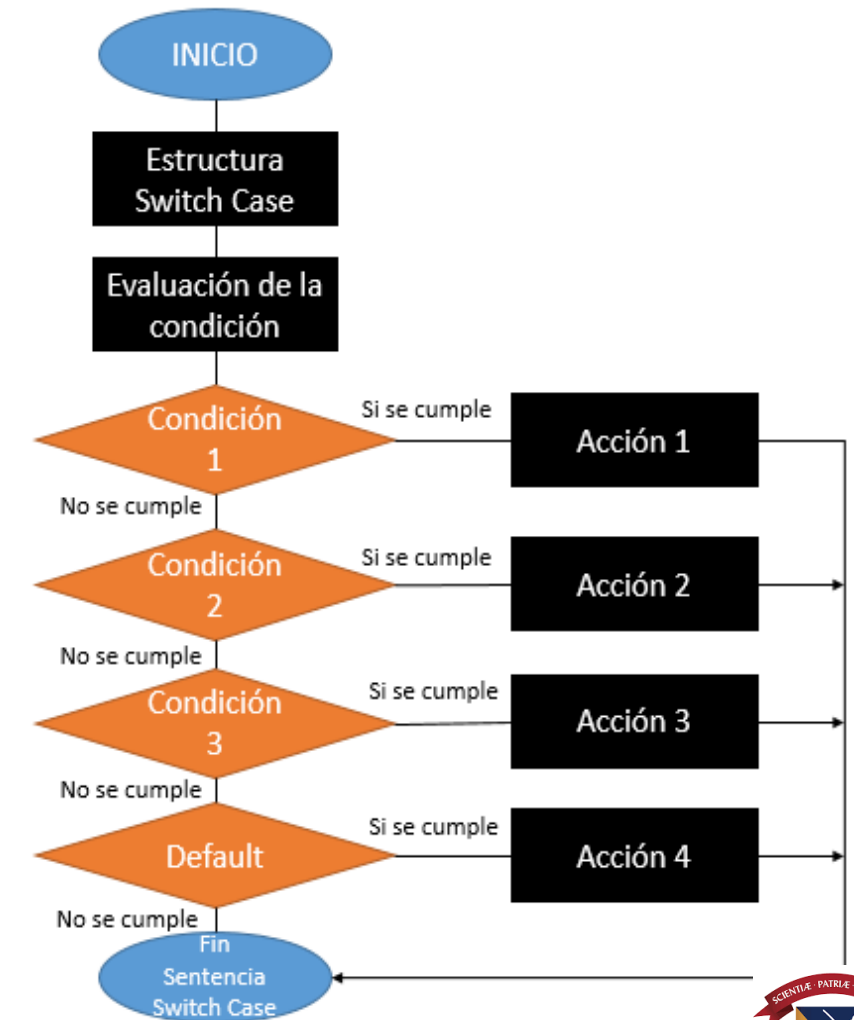


Estructura de programación en C

➤ Estructura SWITCH CASE:

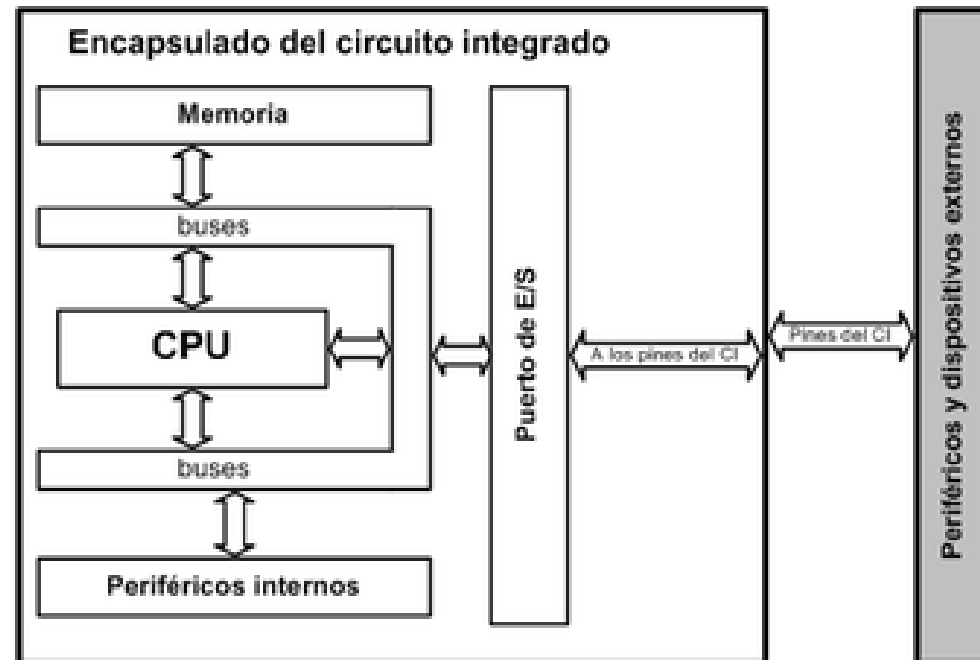
- Se utiliza para evaluar una variable y ejecutar un código según su valor. Cada condición del Case termina en un *break*, y las condiciones irrelevantes se ponen en *default*.

```
int valor; //Declaracion de la Variable
switch(valor)
{
    case 0:      //Codigo que se ejecuta si valor es igual a 0
        break; //Ruptura del codigo para case 0
    case 10:     //Codigo que se ejecuta si valor es igual a 10
        break; //Ruptura del codigo para case 10
    case 20:     //Codigo que se ejecuta si valor es igual a 20
        break; //Ruptura del codigo para case 20
    case 30:     //Codigo que se ejecuta si valor es igual a 30
        break; //Ruptura del codigo para case 30
    default:    //Codigo que se ejecuta si valor es otro valor
        break;
}
```



Arquitectura de los Sistemas embebidos

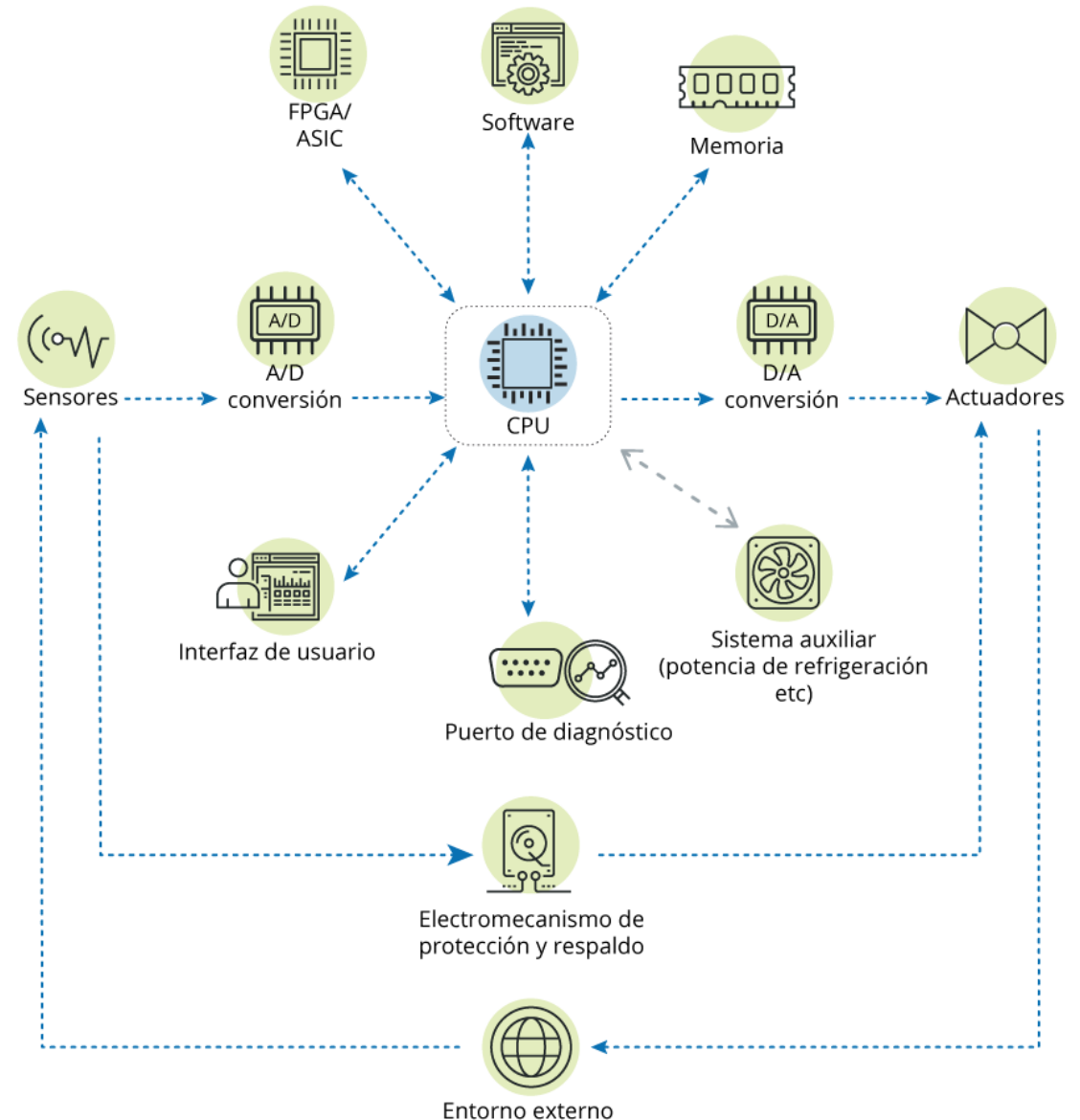
- Los elementos básicos de un microcontrolador son:
 - Microprocesador
 - Periféricos (entrada/salida)
 - Memoria



Arquitectura de los Sistemas embebidos

➤ Algunos de los componentes que hacen parte de los sistemas embebidos son:

- CPU
- Conversores A/D, D/A
- Memoria
- Interfaz de usuario
- Software
- FPGA/ASIC
- Entre otros...



<https://hetpro-store.com/TUTORIALES/microcontrolador/>



Arquitectura de los Sistemas embebidos

➤ Para un sistema embebido específico, como la STM32F407VG, se encuentran componentes como:

- Microcontrolador ARM 32-bit *CortexTM*
- Oscilador de 168 MHZ
- *Memory Flash* de 1 Mbyte
- SRAM de 192+4 Kbyte
- CCM (Core Coupled Memory) de 64 Kbyte
- Interfaz paralela de LCD
- Interfaz paralela a 8-14 bits de cámara a 54 Mbytes/s
- Unidad de cálculo CRC
- RTC: Precisión inferior a segundos y Calendario Hardware
- 3 conversores A/D de 12 bits
- 2 conversores D/A de 12 bits
- 17 temporizadores, 15 de estos a 16 bits y 2 a 32 bits, con funciones IC/OC/PWM
- 140 I/O con capacidad de interrupción
- 15 interfaces de comunicación
- 4 LEDs de uso general
- Conexiones para periféricos de entrada y salida



QUIZ

Escriba un código en lenguaje c, que entregue el promedio de los elementos de una matriz, con los siguientes valores:

5	4	3
1	2	6
3	7	5

TAREA

Instalar el software de programación keil según las indicaciones dadas en:

<https://www.youtube.com/watch?v=NZghf4fi-j8&t=565s>

