



**PUCP**

# PROGRAMACIÓN DE MICROCONTROLADORES ARM

*CETAM - PUCP*



# Sesión 3 - 09/07/2023:

- Git, Github - Aplicaciones
- Sistemas y Circuitos Digitales
- Microcontroladores ARM
- Ejemplos de aplicación



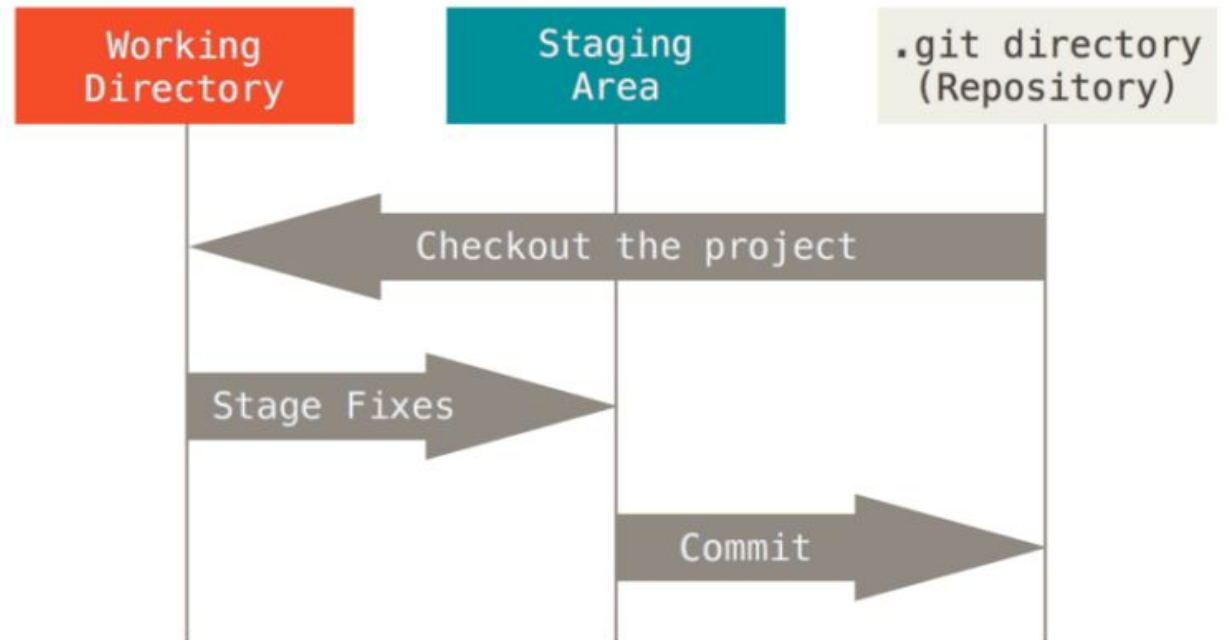
# Sistema de control de versiones GIT

*Programación de microcontroladores ARM - Sesión 3*

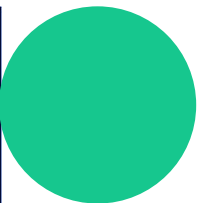
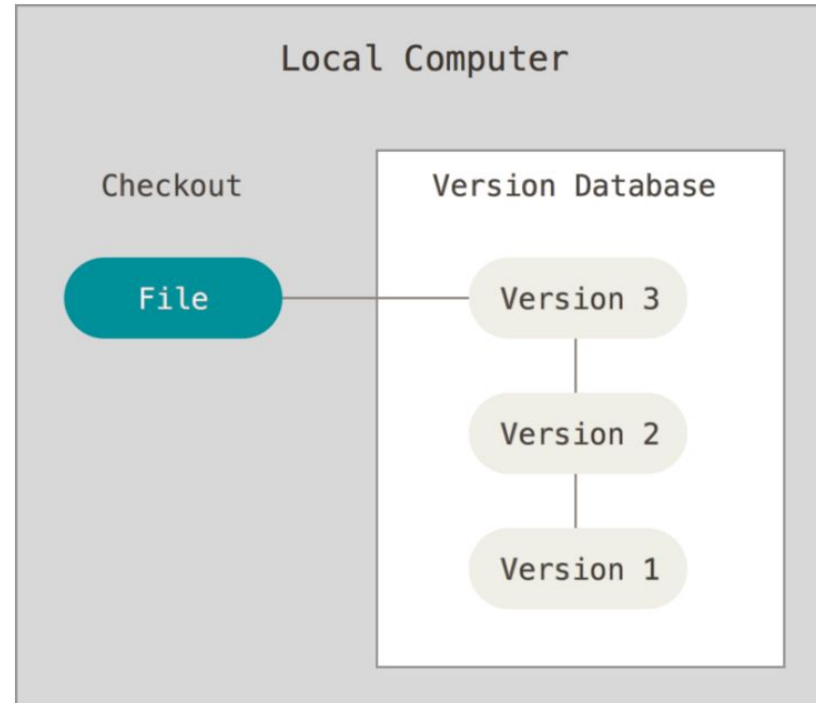


# ¿Qué es GIT?

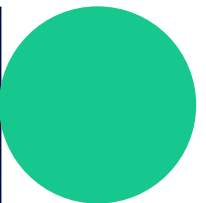
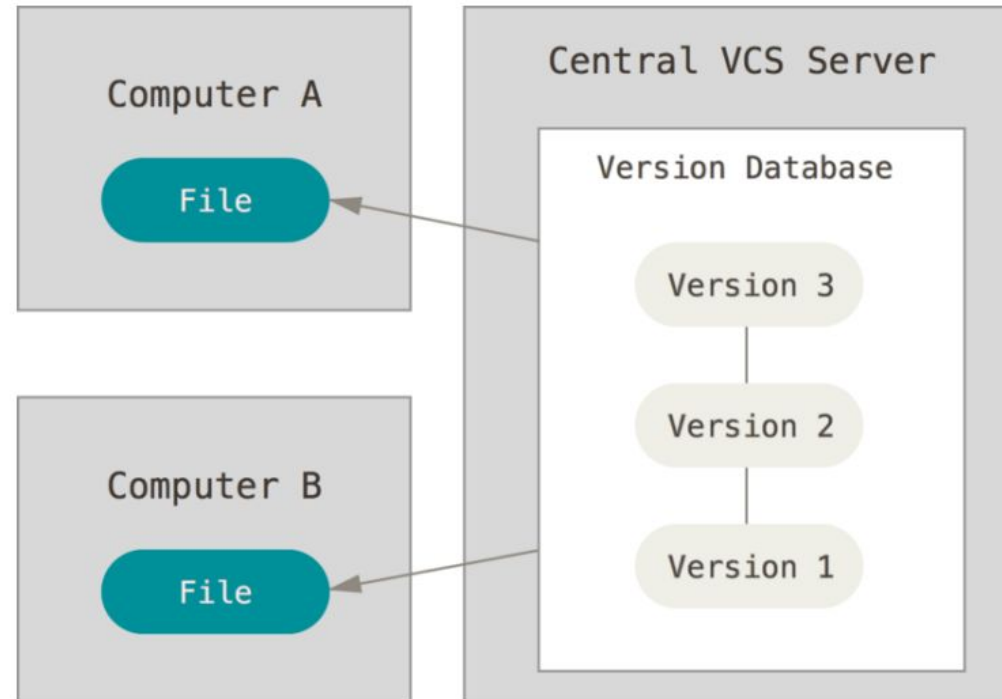
Es un sistema de control de versiones que permite mantener el historial de cambios entre los archivos de un determinado proyecto. Esta característica permite que los cambios realizados puedan ser revertidos en caso se detecten errores.



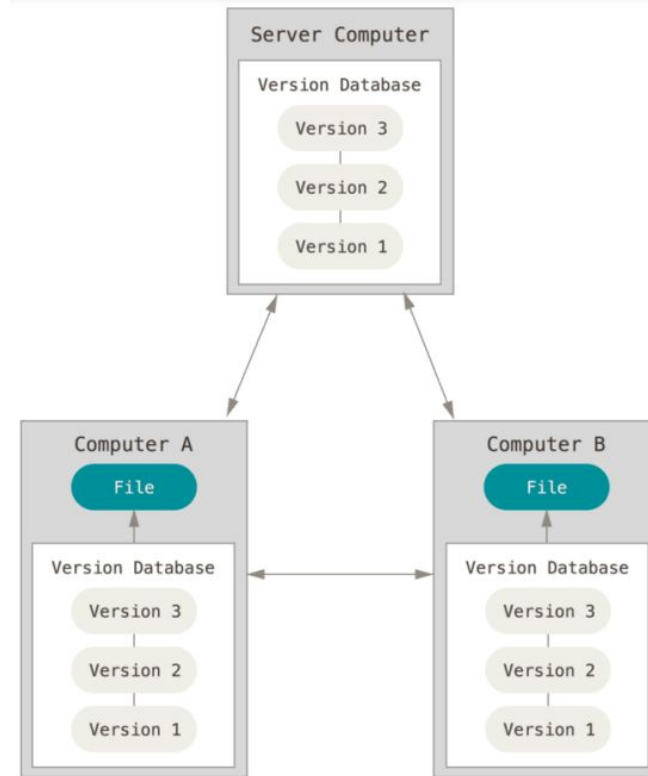
# Sistema local de control de versiones



# Sistema centralizado de control de versiones



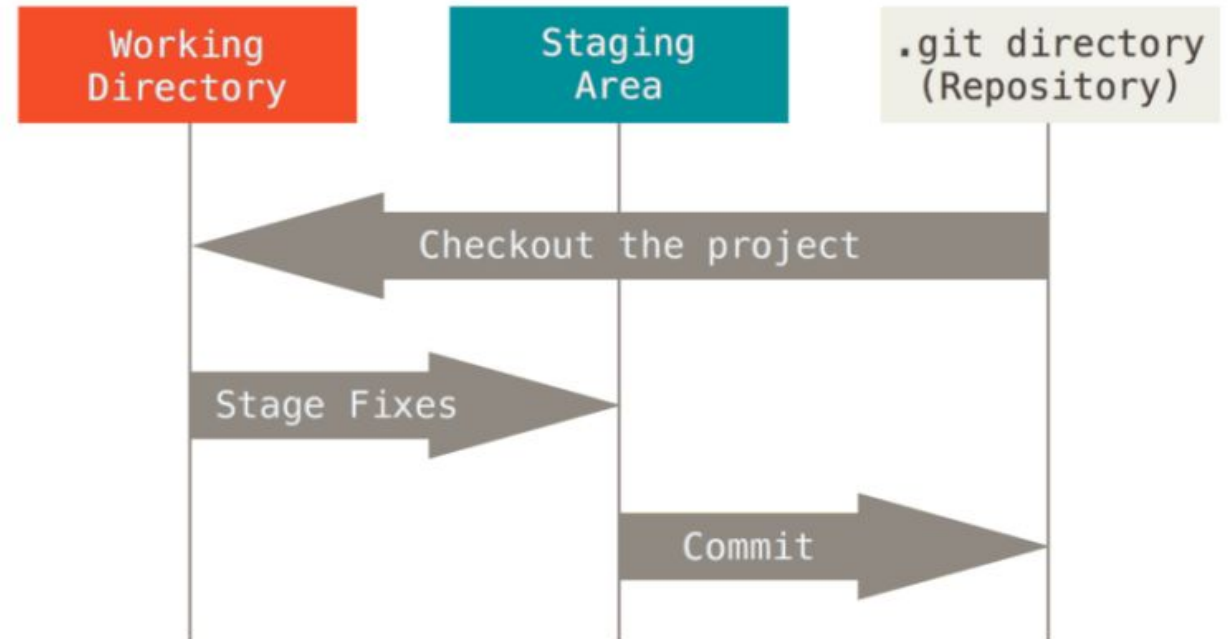
# Sistema distribuido de control de versiones





# Estados de un repositorio GIT

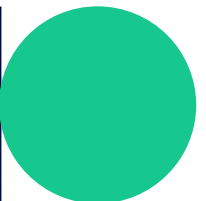
- Archivos no trackeados.- Son archivos de los cuales no se lleva un registro de sus versiones y son ignorados por el sistema
- Archivos agregados.- Son archivos colocados en el área de carga, poseen modificaciones pero aún no se encuentran trackeados
- Archivos trackeados.- Son archivos del cual se lleva el registro de sus versiones; todos los archivos agregados pasan a este estado luego de un "commit".



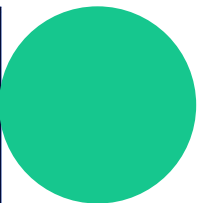
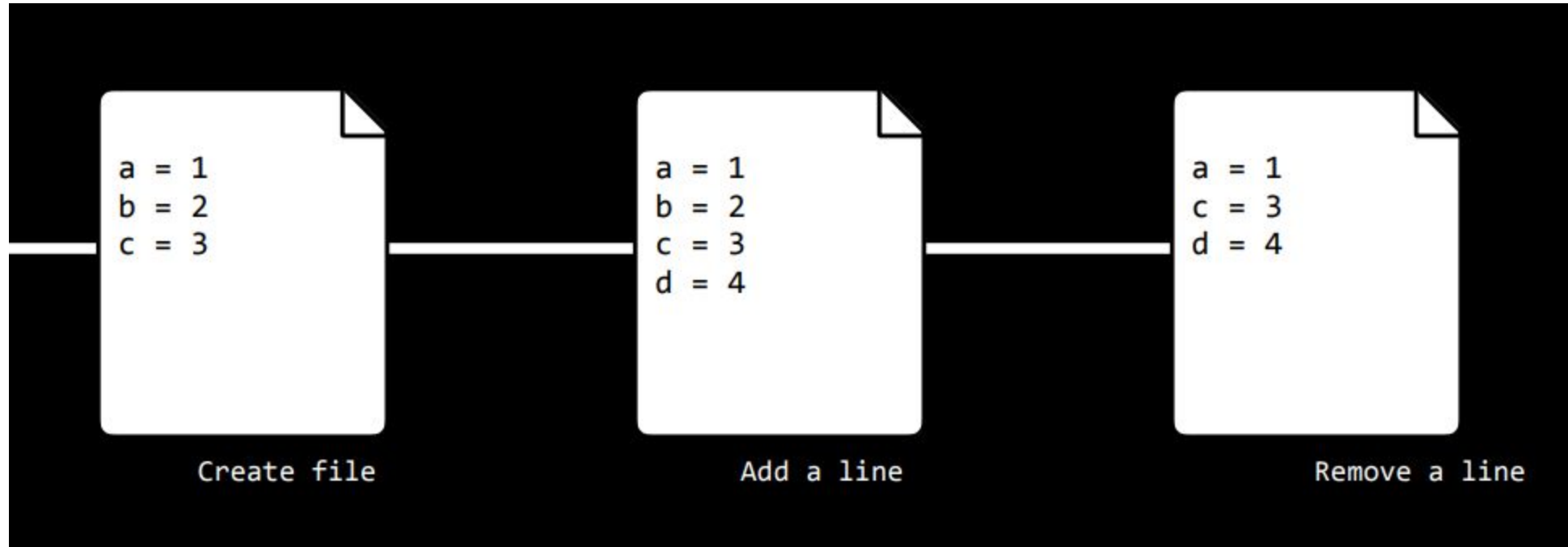
# Características de GIT

Dentro de las características de git tenemos lo siguiente:

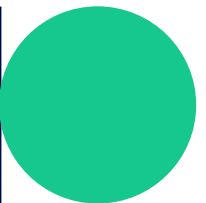
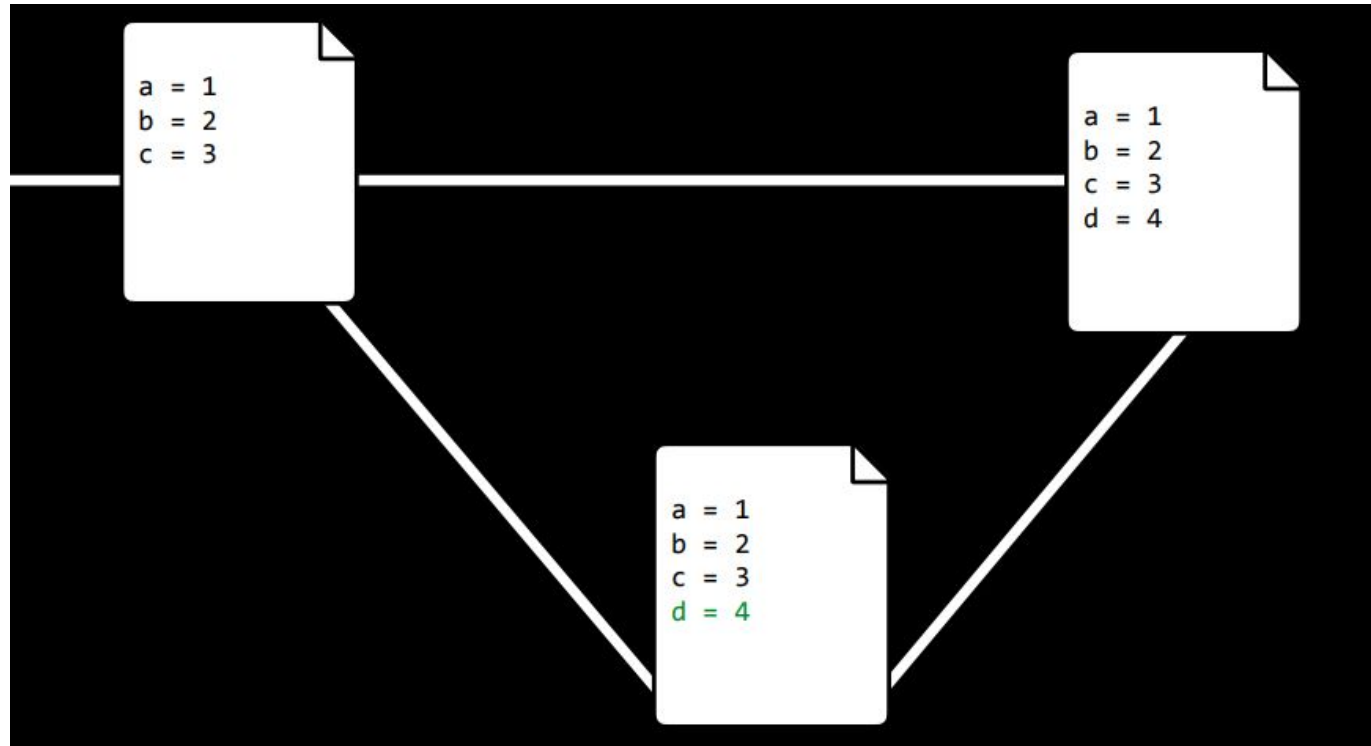
- Verificar los cambios entre diferentes versiones
- Sincronizar los cambios entre diferentes desarrolladores
- Crear ramas de experimentación del proyecto
- Regresar a versiones anteriores del proyecto



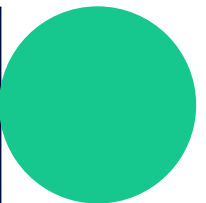
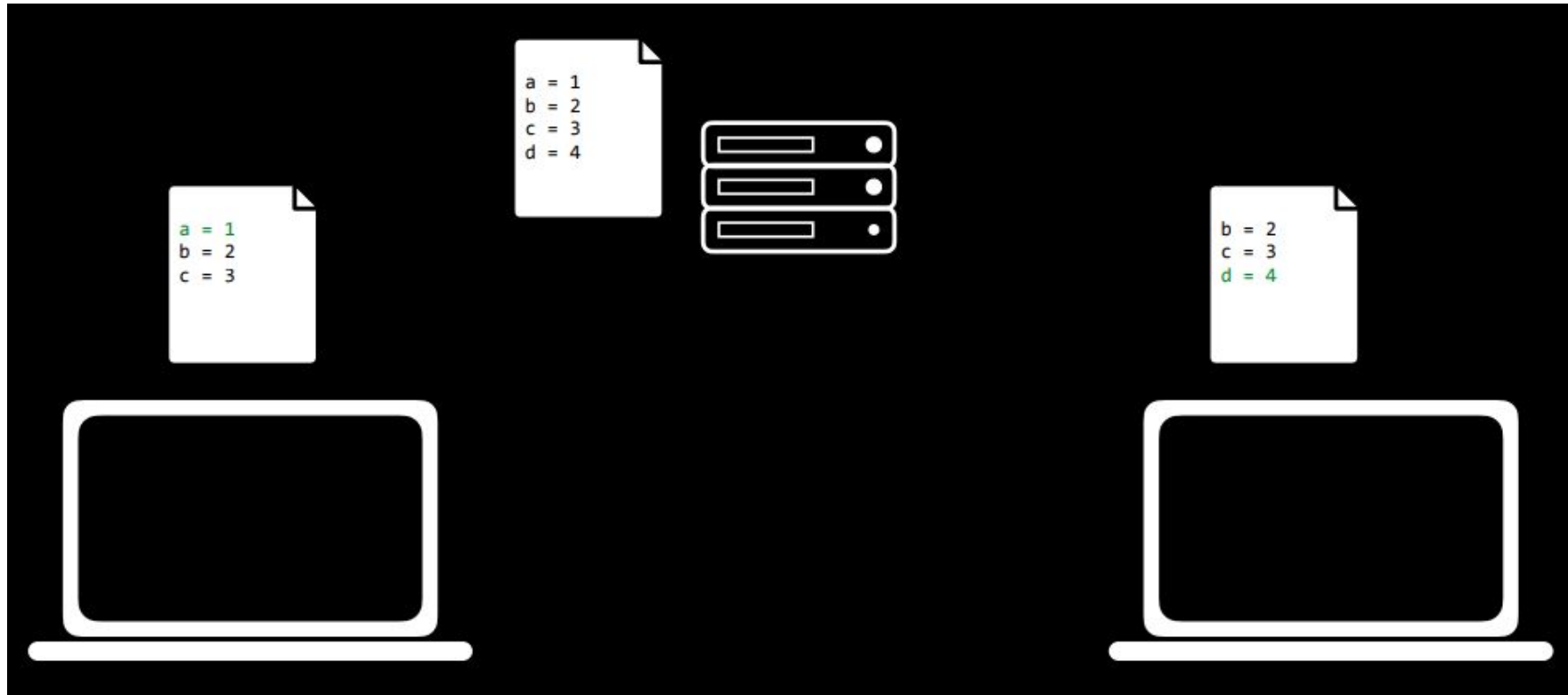
# Verificar los cambios de código



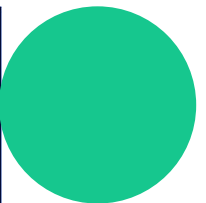
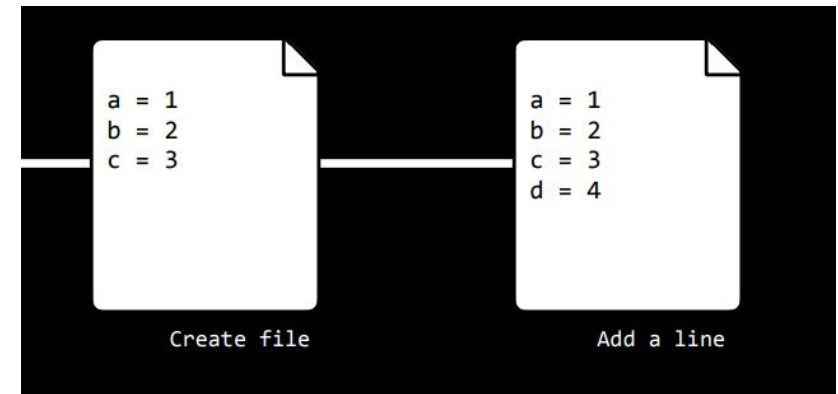
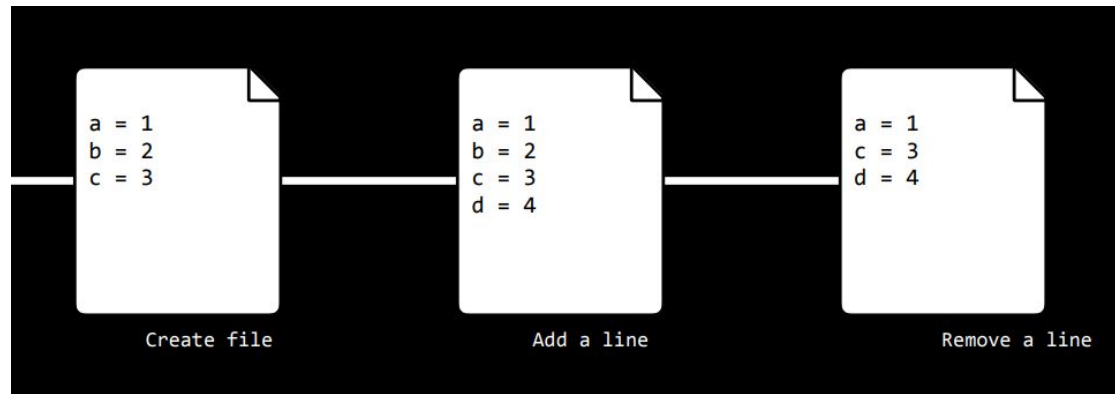
# Verificar cambios en el código sin perder la versión original



# Sincronizar cambios entre diferentes desarrolladores



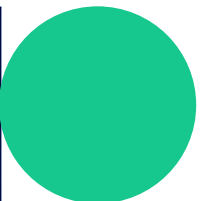
# Regresar a versiones anteriores del código

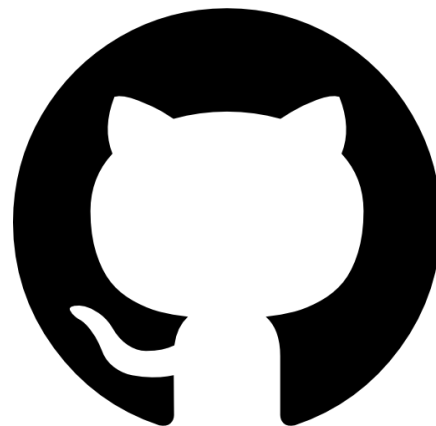


# Principales aplicaciones de Git

Dentro de las características de git tenemos lo siguiente:

- Verificar los cambios entre diferentes versiones
- Sincronizar los cambios entre diferentes desarrolladores
- Crear ramas de experimentación del proyecto
- Regresar a versiones anteriores del proyecto





**GitHub**

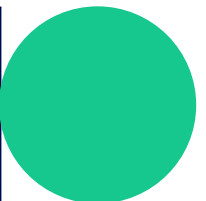
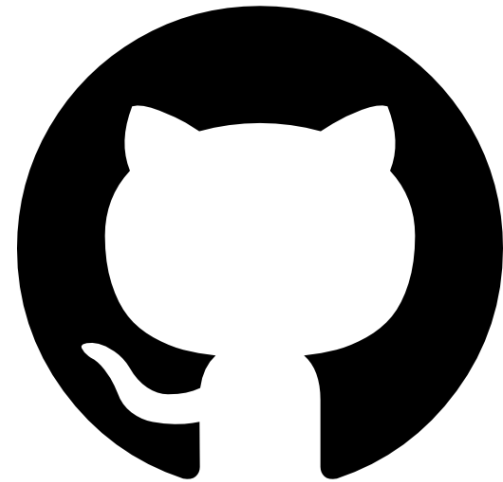
*Programación de microcontroladores ARM - Sesión 3*





# ¿Qué es Github?

Github es un servidor de almacenamiento para repositorios de git, en donde se almacenan los diferentes proyectos realizados. Además posee un herramientas gráficas para revisar las diferentes versiones de un repositorio



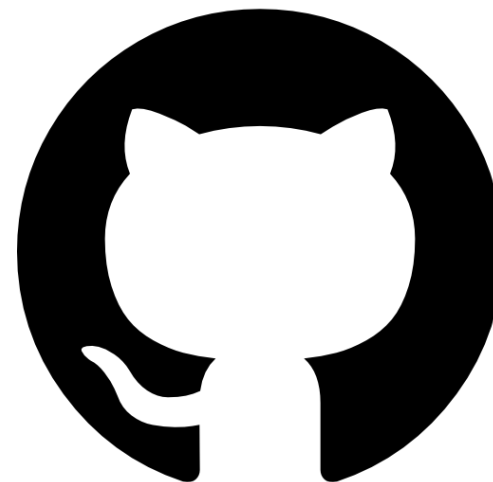
# Repositorios en Github

Verificando algunos desarrolladores  
notables:

- <http://rafalab.github.io/>
- <https://github.com/dmalan>

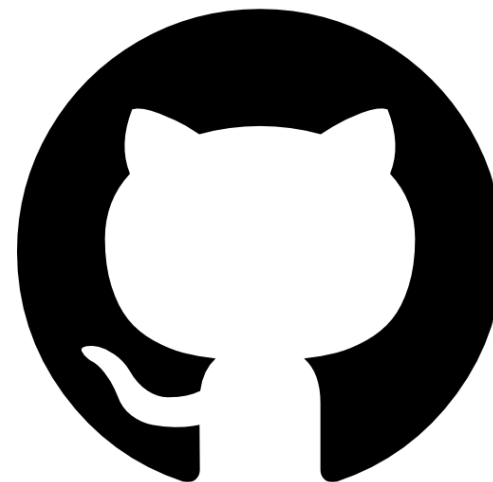
Repositorio de ejemplo:

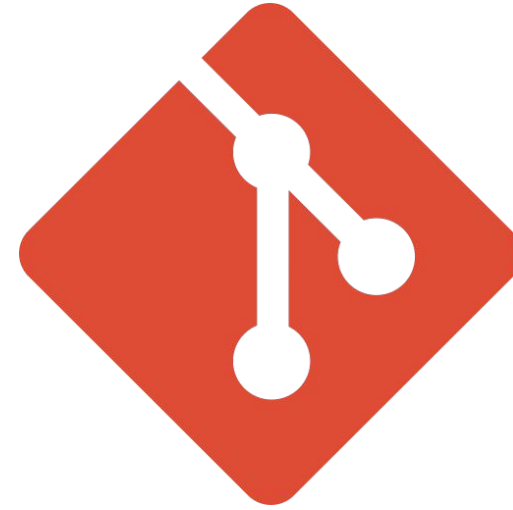
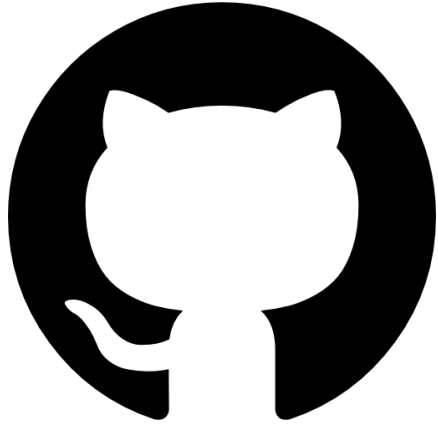
- <https://github.com/emwhbr/stm32-bluepill>



# Git push y Git clone

Los comandos `Git Push` y `Git Clone` permiten extraer y subir los cambios realizados a un repositorio de github. Cada repositorio posee en su configuración un origen remoto al cual se hace la petición al momento de clonarlo o actualizarlo.





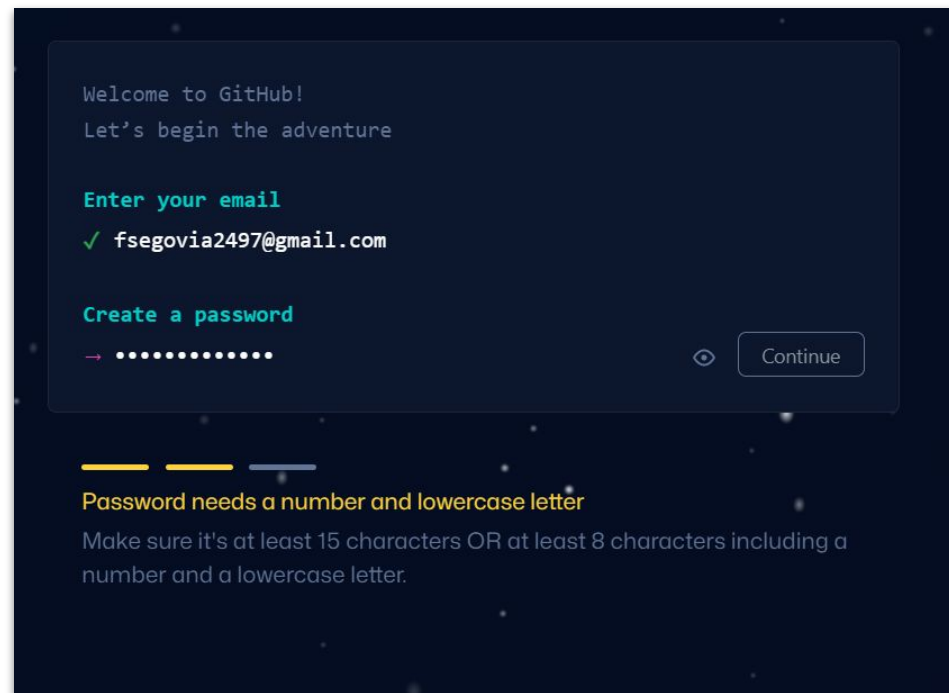
# Aplicaciones Git y Github

*Programación de microcontroladores ARM - Sesión 3*



# Crear una cuenta en github y generar el token

- Ingresar nuestro Email y crear un password

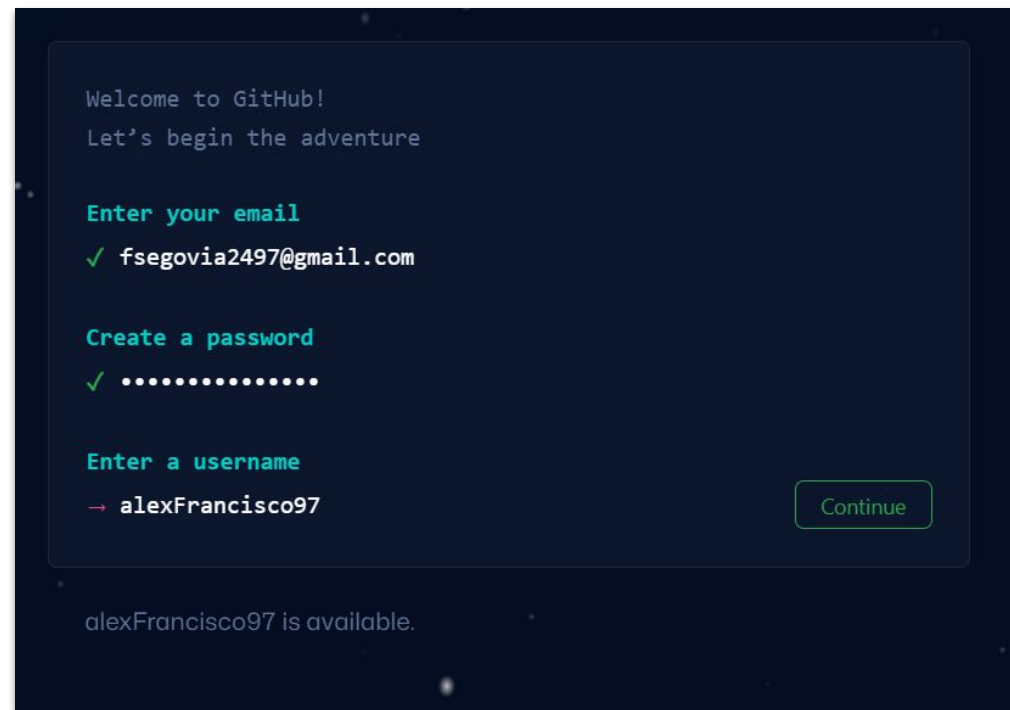


The screenshot shows the GitHub account creation page. It has a dark blue background with a light blue box for the form. The text inside the box says: "Welcome to GitHub! Let's begin the adventure". Below this, there are two sections: "Enter your email" and "Create a password". The email field is filled with "fsegovia2497@gmail.com" and has a green checkmark. The password field is filled with dots and has a red arrow pointing to it. To the right of the password field is a "Continue" button. Below the form, there is a progress bar with three segments, the first two of which are yellow. Below the progress bar, there is a warning message: "Password needs a number and lowercase letter". Below the warning message, there is a note: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter."

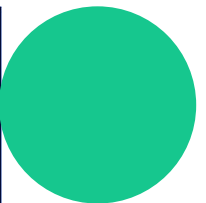


# Crear una cuenta en github y generar el token

- Ingresar nuestro Email, generar una contraseña y un nombre de usuario

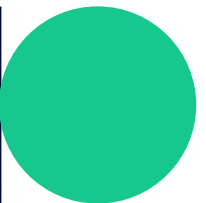
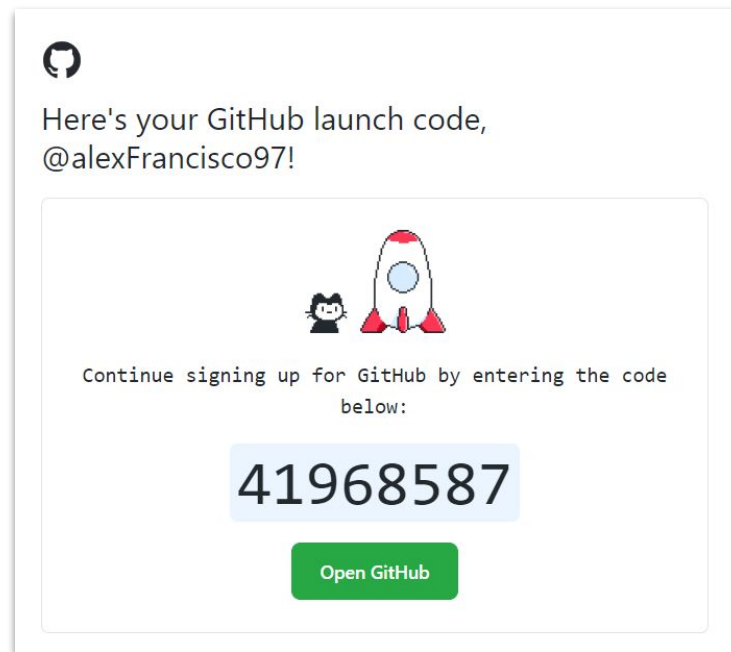


The screenshot shows the GitHub account creation interface. It has a dark blue background with white text. At the top, it says "Welcome to GitHub!" and "Let's begin the adventure". Below this, there are three sections: "Enter your email" with a green checkmark and the email "fsegovia2497@gmail.com"; "Create a password" with a green checkmark and a series of dots; and "Enter a username" with a red arrow and the username "alexFrancisco97". A green "Continue" button is located to the right of the username field. At the bottom, it says "alexFrancisco97 is available."



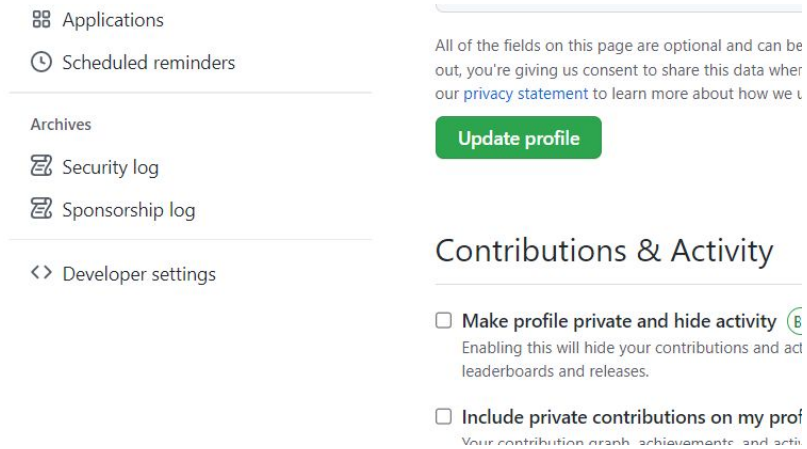
# Crear una cuenta en github y generar el token

- Confirmar el correo electrónico



# Crear una cuenta en github y generar el token

- En configuración ingresar a 'Opciones de desarrollador' y 'Generar nuevo token'



Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

## Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the GitHub API.

commits\_agosto — admin:pkg\_key, delete\_packages, delete\_repo, project, repo, write\_packages

Last used within the last week

Delete

Expires on Tue, Sep 20 2022.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.



# Crear y trackear un repositorio

- `git init`            `#Inicializa un repositorio`
- `git log`            `#Muestra el historial de versiones del repo`
- `git status`        `#Muestra el estado de los archivos en el repo`
- `git add`            `#Agrega archivos al área de pre-carga`
- `git commit`        `#Crea una nueva versión del repositorio`
- `git clone`        `#Crea una copia local de un repositorio remoto`

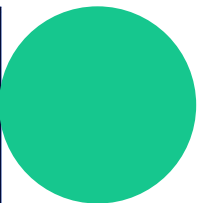
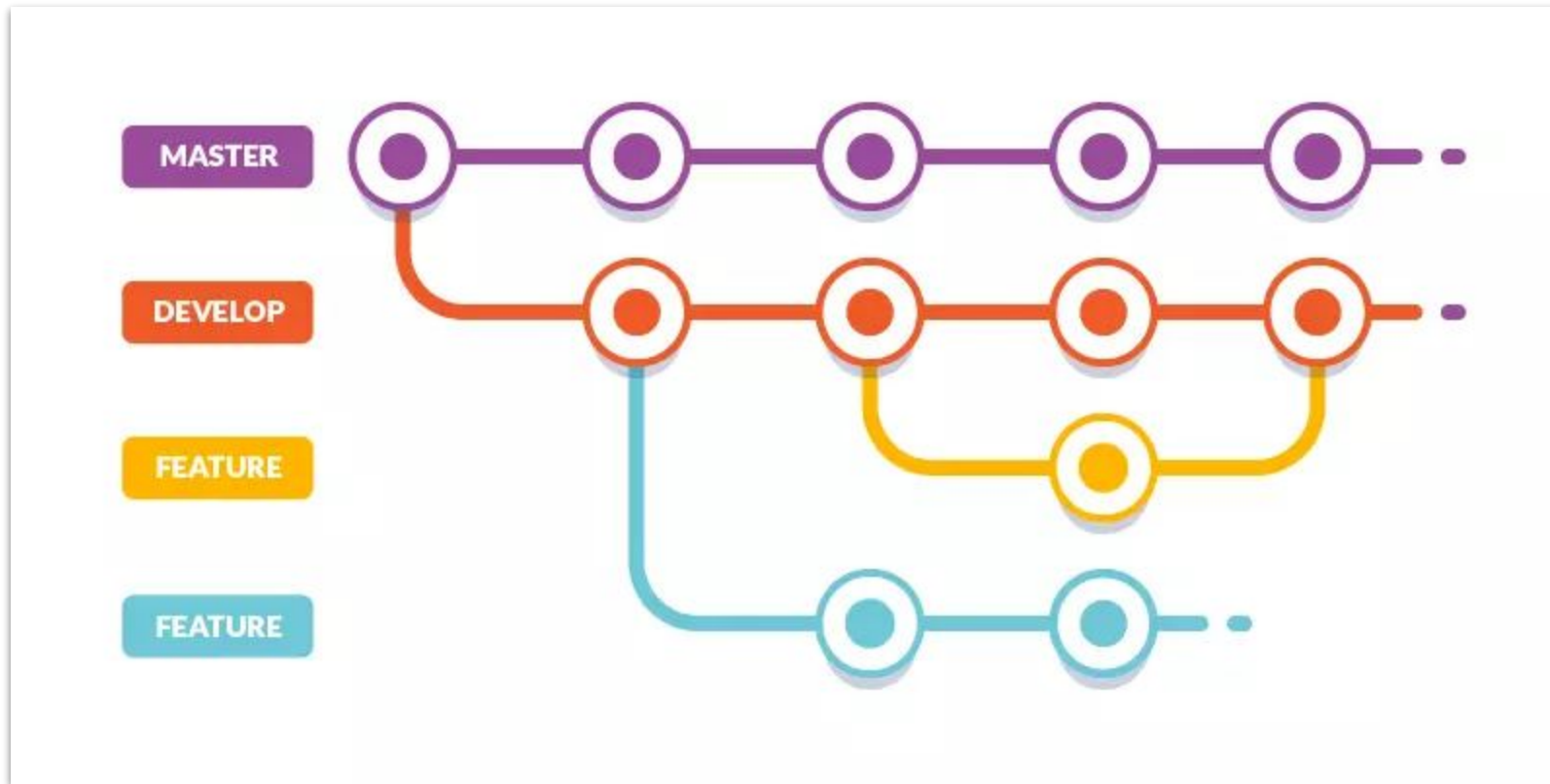


# Comandos más comunes en Git

- `git log`           #Muestra el historial de versiones del repo
- `git status`       #Muestra el estado de los archivos en el repo
- `git clone`        #Crea una copia local de un repositorio remoto
- `git branch`       #Permite crear diferentes ramas en el repo
- `git checkout`    #Permite visualizar las diferentes versiones
- `git reset`        #Eliminar los cambios posteriores
- `git merge`        #Permite combinar dos ramas diferentes



# Ramas en un repositorio

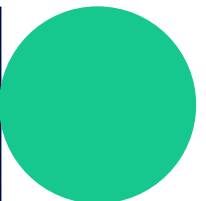
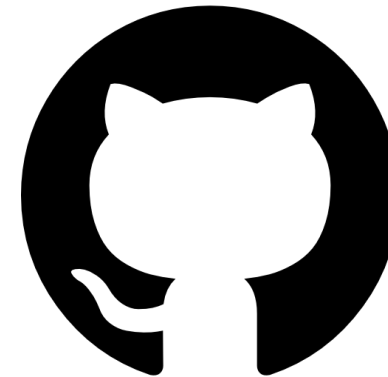
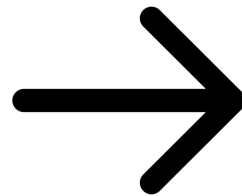


# Subir cambios a un repositorio remoto

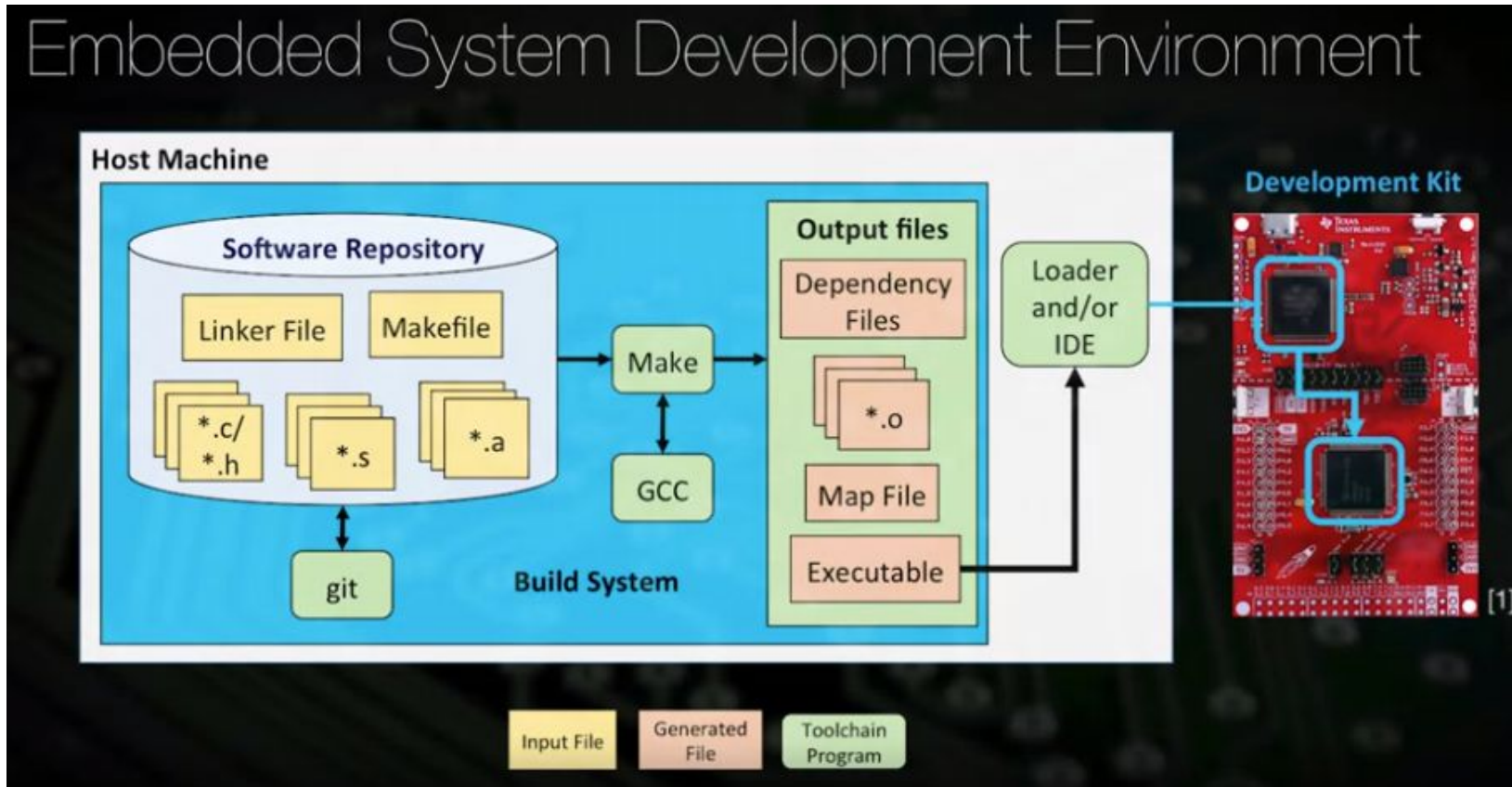
- `git remote` `#Visualiza las fuentes remotas`
- `git remote add <url>` `#Permite agregar una fuente remota al repo`
- `git push origin master` `#Empuja los cambios locales al repo remoto`



git

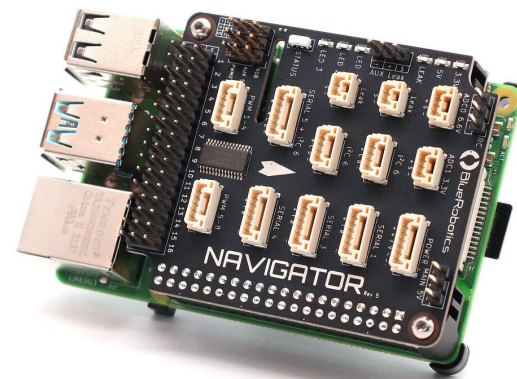
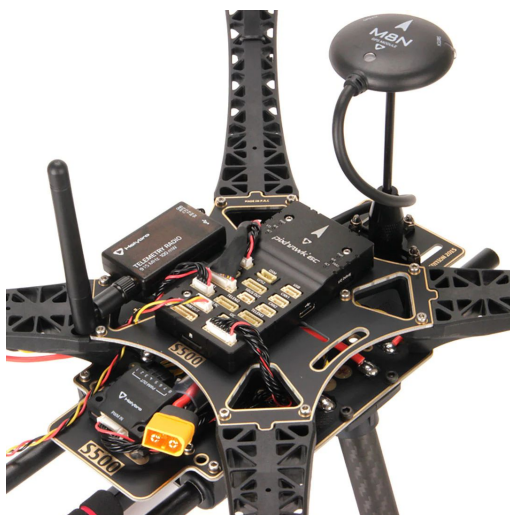


# Desarrollo embebido



# Ejemplo de repositorios de sistemas embebidos

- AUTOPILOT - <https://github.com/PX4/PX4-Autopilot>
- ARDUPILOT - <https://github.com/ArduPilot/ardupilot>
- ARDUSUB - <https://github.com/bluerobotics/ardusub>
- Especificaciones ARDUPILOT: <https://ardupilot.org/copter/docs/common-pixhawk-overview.html>



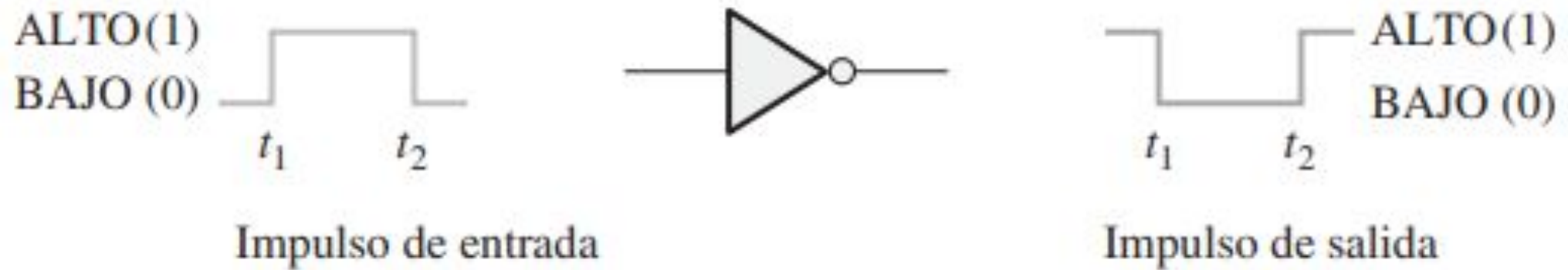


# Sistemas y circuitos digitales

*Programación de microcontroladores ARM - Sesión 3*




# Circuito Inversor






# Compuerta lógica AND

BAJO (0) —  
BAJO (0) —



BAJO (0)

BAJO (0) —  
ALTO (1) —



BAJO (0)

ALTO (1) —  
BAJO (0) —



BAJO (0)

ALTO (1) —  
ALTO (1) —



ALTO (1)

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	0
0	1	0
1	0	0
1	1	1
1 = ALTO, 0 = BAJO		

# Compuerta lógica OR

BAJO (0) —  
BAJO (0) —



BAJO (0)

BAJO (0) —  
ALTO (1) —



ALTO (1)

ALTO (1) —  
BAJO (0) —



ALTO (1)

ALTO (1) —  
ALTO (1) —



ALTO (1)

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	0
0	1	1
1	0	1
1	1	1
1 = ALTO, 0 = BAJO		

# Compuerta lógica NAND

BAJO (0) —  
BAJO (0) —



ALTO(1)

BAJO (0) —  
HIGH (1) —




ALTO(1)

ALTO(1) —  
BAJO (0) —



ALTO(1)

ALTO(1) —  
ALTO(1) —



BAJO (0)

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	1
0	1	1
1	0	1
1	1	0
1 = ALTO, 0 = BAJO		

# Compuerta lógica NOR

BAJO (0) —  
BAJO (0) —



ALTO (1)

BAJO (0) —  
ALTO (1) —



BAJO (0)

ALTO (1) —  
BAJO (0) —



BAJO (0)

ALTO (1) —  
ALTO (1) —



BAJO (0)

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	1
0	1	0
1	0	0
1	1	0
1 = ALTO, 0 = BAJO		

# Compuerta OR-EXCLUSIVA

BAJO (0) —  
BAJO (0) —




BAJO (0)

BAJO (0) —  
ALTO (1) —



ALTO (1)

ALTO (1) —  
BAJO (0) —



ALTO (1)

ALTO (1) —  
ALTO (1) —

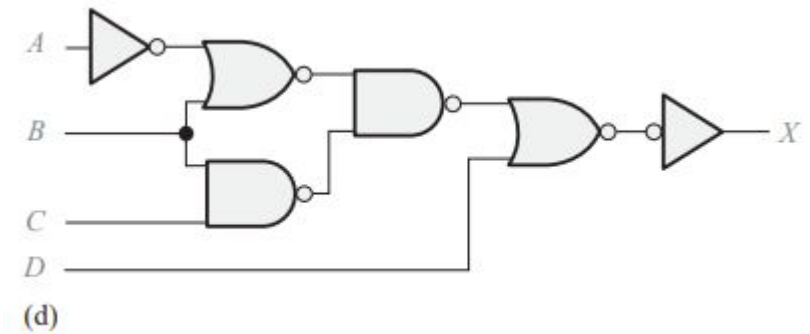
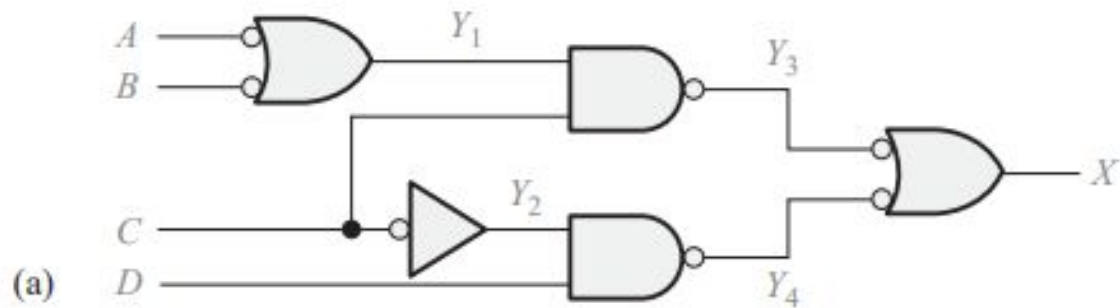


BAJO (0)

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	0
0	1	1
1	0	1
1	1	0

# Lógica combinacional

Encontrar la relación lógica entre  $A$ ,  $B$ ,  $C$  y  $D$  con el resultado  $X$



# Circuito sumador de 2 bits

BAJO (0) —  
BAJO (0) —



ALTO(1)

BAJO (0) —  
ALTO(1) —



BAJO (0)

ALTO(1) —  
BAJO (0) —



LOW (0)

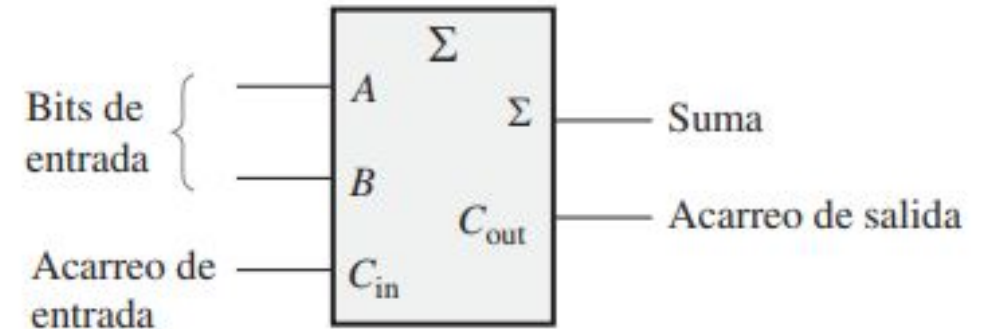
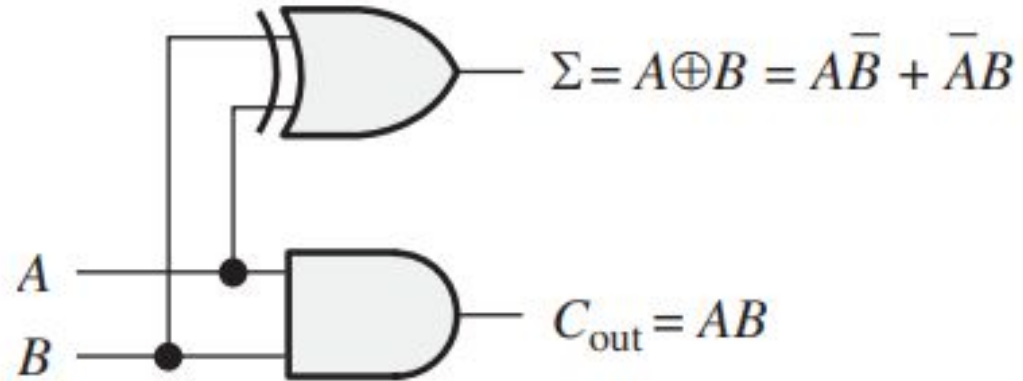
ALTO(1) —  
ALTO(1) —



ALTO(1)

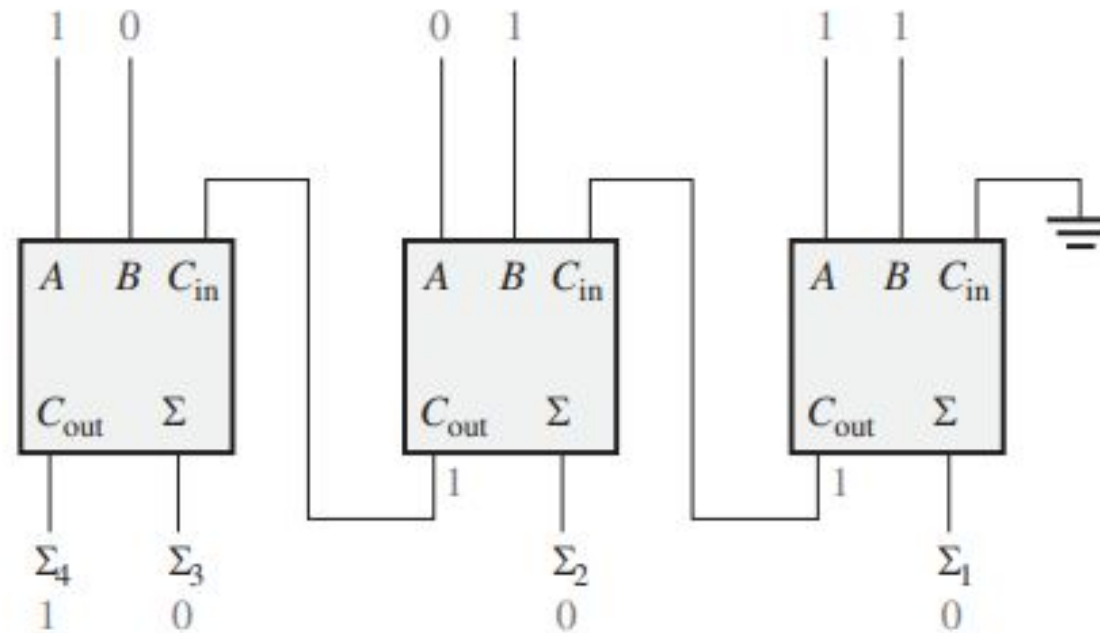
Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	1
0	1	0
1	0	0
1	1	1

# Circuito sumador de 2 bits

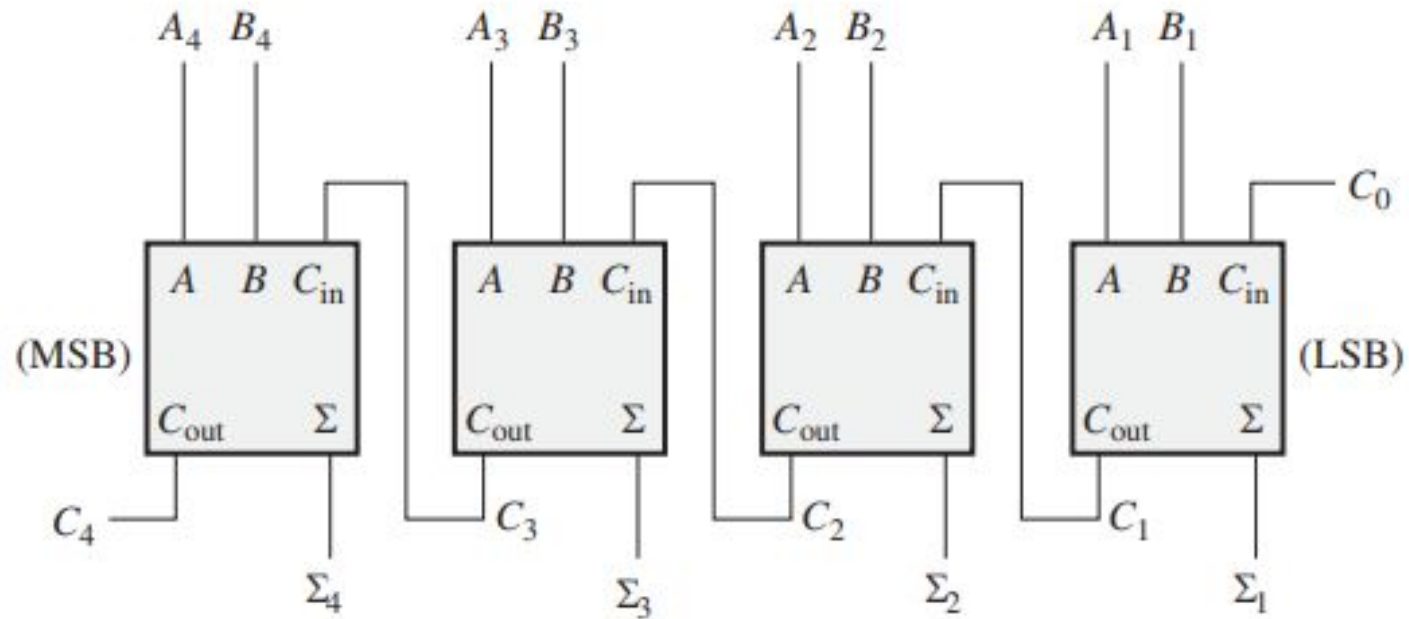




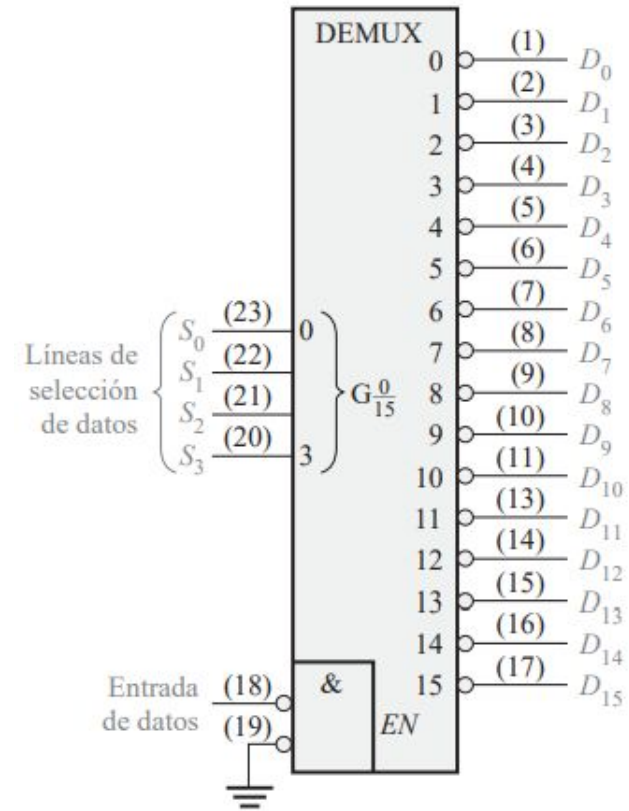
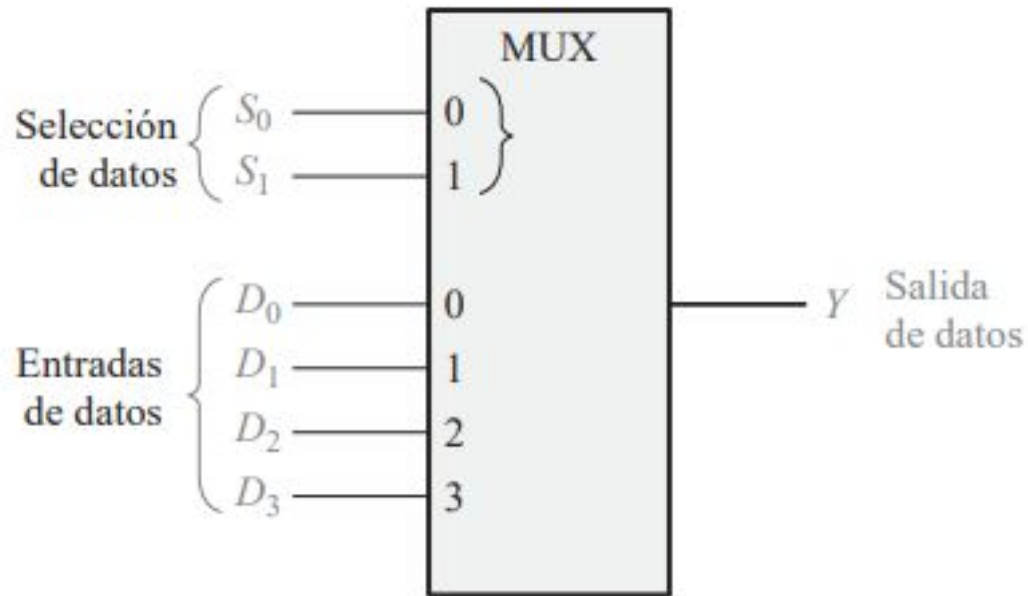
# Circuito sumador de 3 bits



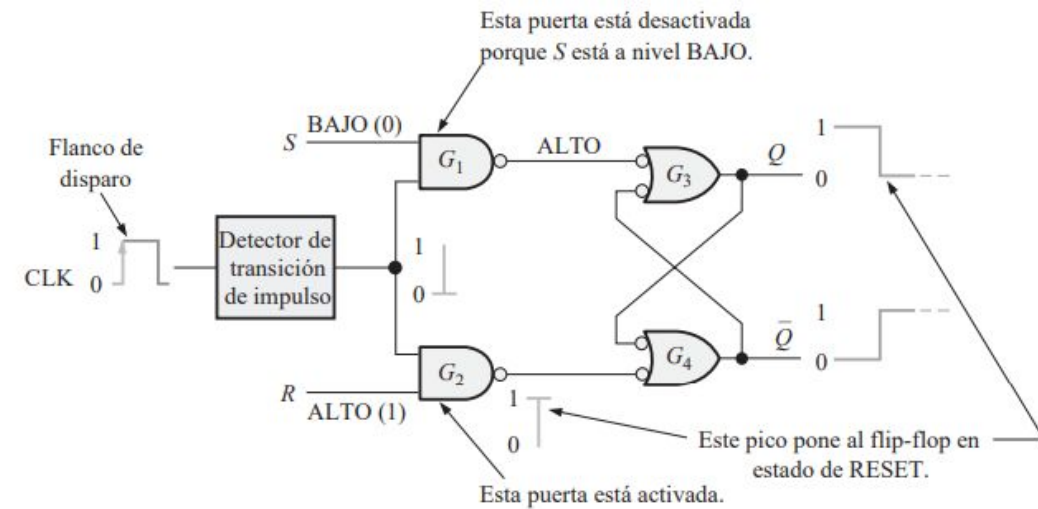
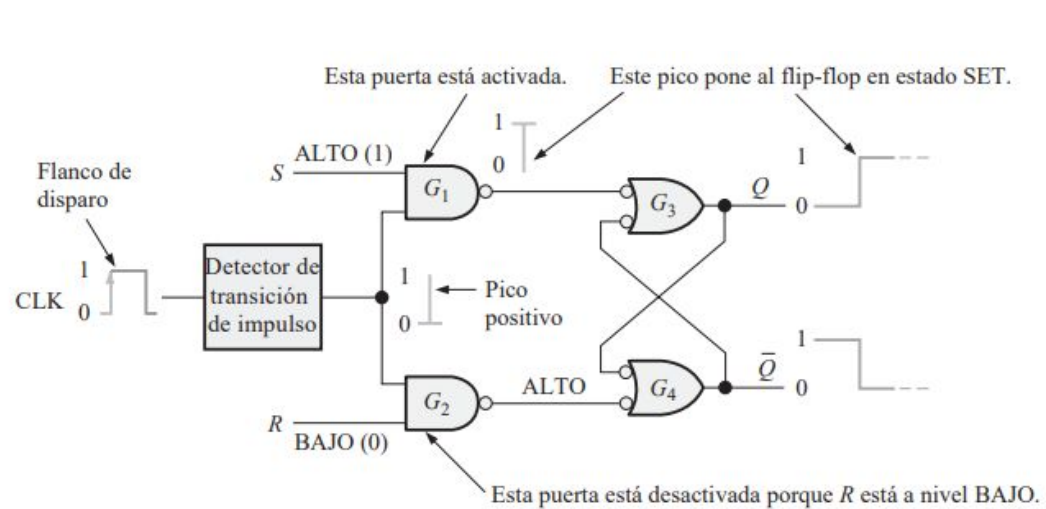
# Circuito sumador de 4 bits



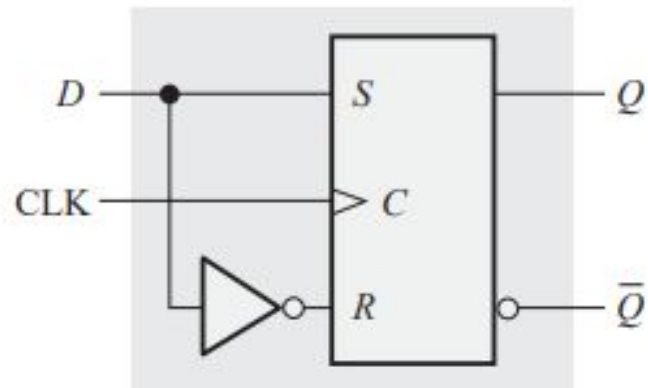
# Multiplexores y Demultiplexores



# Latches y Flip-Flops

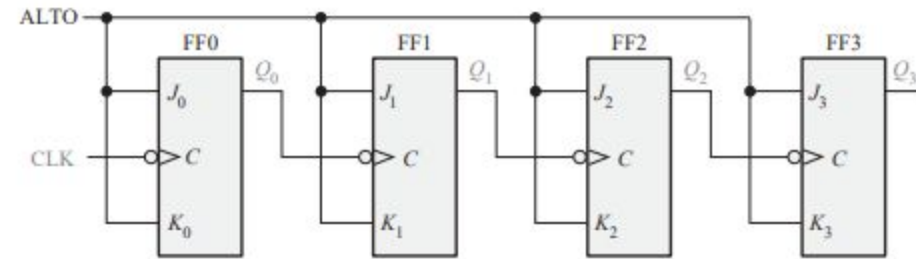


# Latches y Flip-Flops

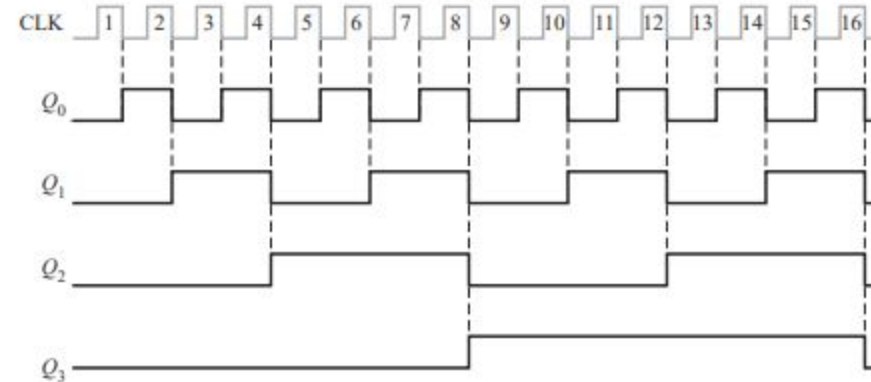


Entradas		Salidas		Comentarios
$D$	$CLK$	$Q$	$\bar{Q}$	
1	$\uparrow$	1	0	SET (almacena un 1)
0	$\uparrow$	0	1	RESET (almacena un 0)
$\uparrow$ = transición del reloj de nivel BAJO a nivel ALTO				

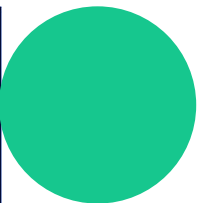
# Contadores



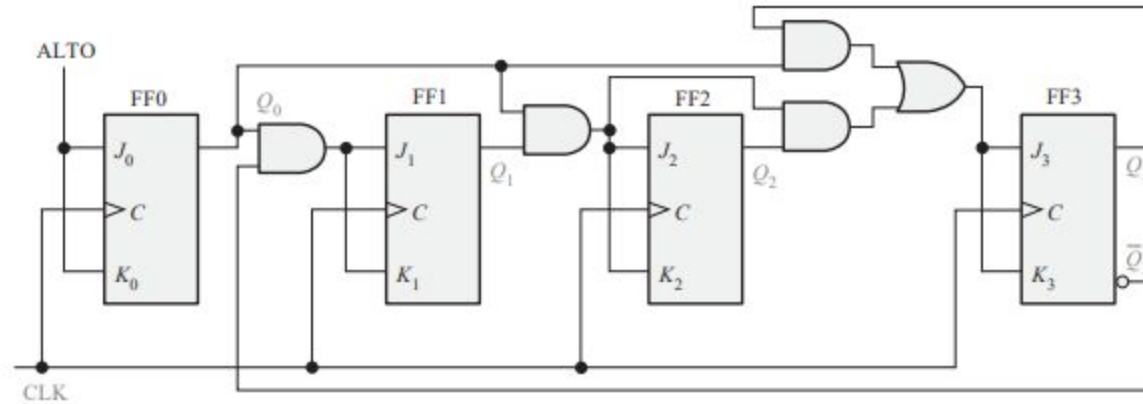
(a)



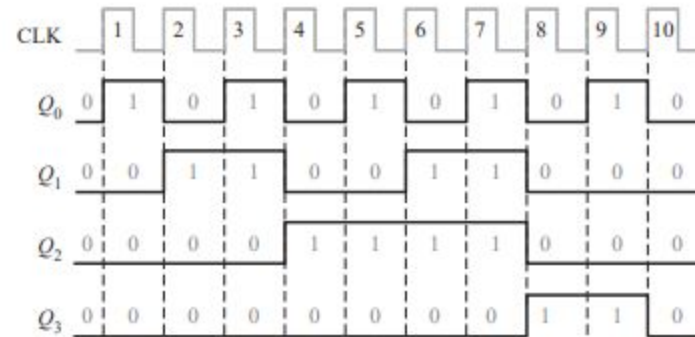
(b)



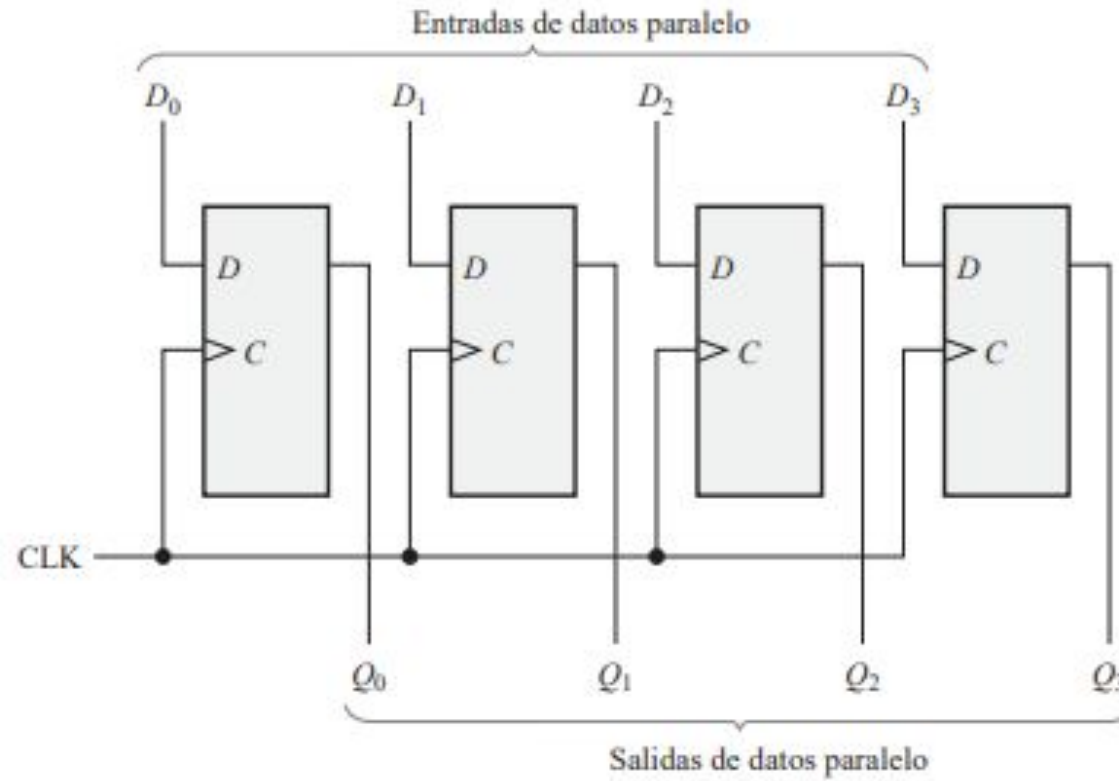
# Contadores



**FIGURA 8.17** Contador de décadas BCD síncrono.

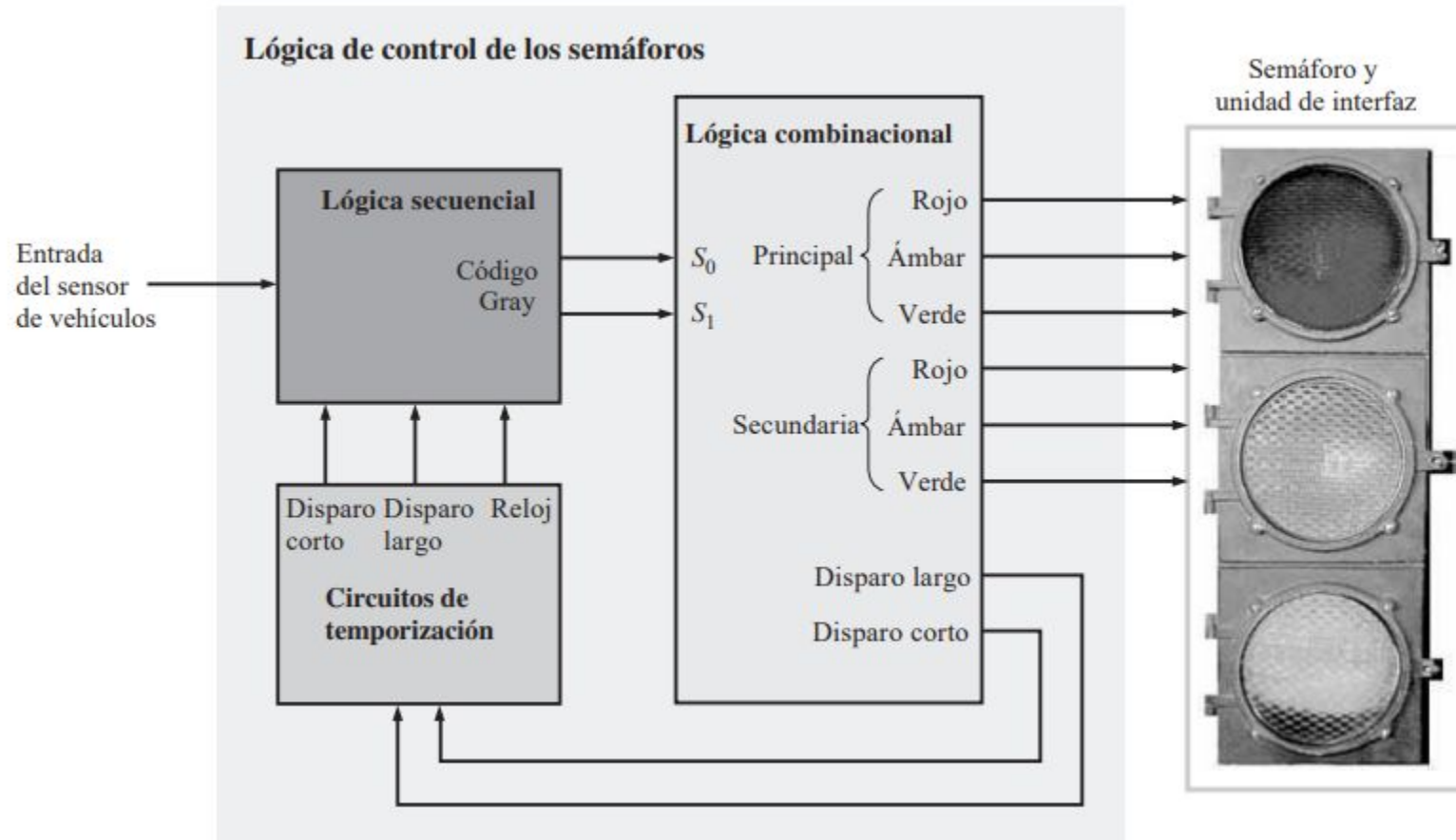


# Registros





# Máquinas de estado



**arm** Developer

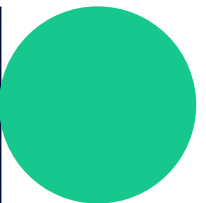
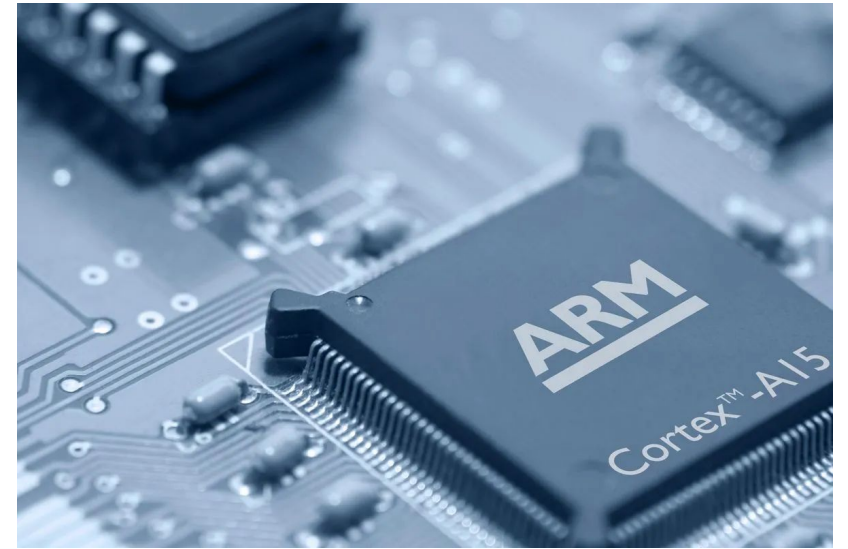
# Microcontroladores ARM

*Programación de microcontroladores ARM - Sesión 3*



# ¿Qué es un procesador ARM?

Un procesador ARM es un sistema digital programable que ejecuta instrucciones lógicas utilizando instrucciones RISC, a diferencia de procesadores INTEL o AMD que utilizan instrucciones CISC



# Instrucciones CISC

Las instrucciones CISC tienen la siguiente estructura :

- Código de operación
- Operandos fuente (OP1, OP2)
- Operando destino o resultado (OPd)
- Dirección de la instrucción siguiente (IS)



# Instrucciones RISC

Las instrucciones RISC tienen la siguiente estructura :

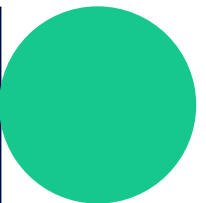
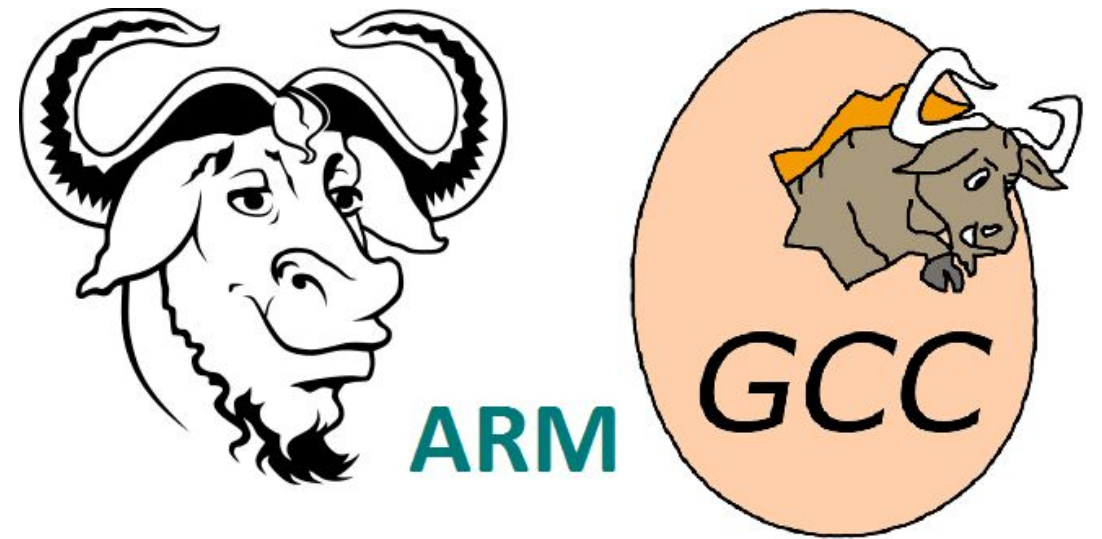
- Código de operación (CO)
- Operandos fuente (OP1, OP2)
- Operando destino o resultado (OPd)

CO	OP1	OP2	OPd
----	-----	-----	-----



# GNU ARM y GNU GCC

- GNU - GCC .- Es la base del compilador para el lenguaje de programación C y traduce hacia el código ensamblador para ordenadores con arquitectura x86, x64, etc
- GNU - ARM .- Utiliza el set de instrucciones RISC y traduce el lenguaje C para dicha estructura



# Cortex - A, Cortex - R, Cortex - M

## Cortex - A

Highest performance

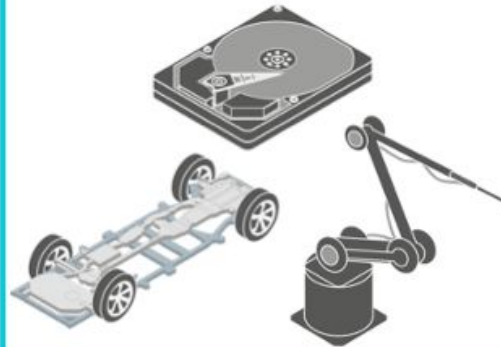
Optimised for  
rich operating systems



## Cortex - R

Fast response

Optimised for  
high performance,  
hard real-time applications



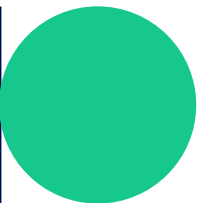
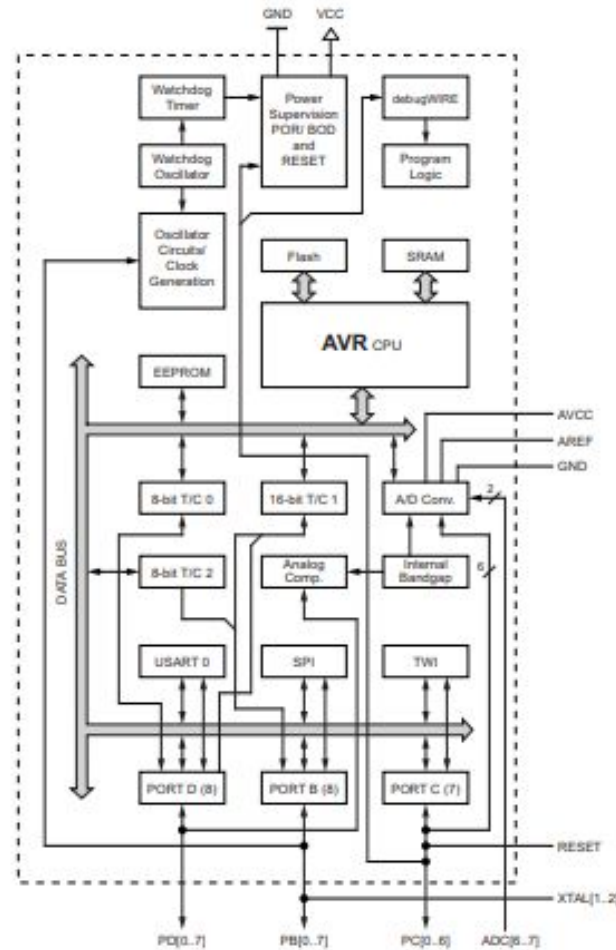
## Cortex - M

Smallest/lowest power

Optimised for  
discrete processing and  
microcontrollers

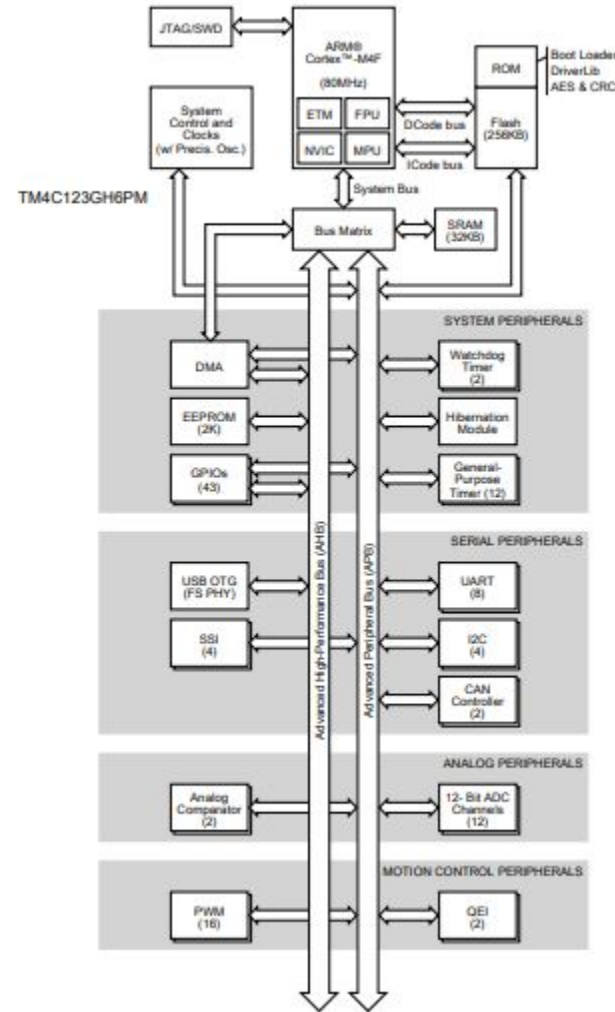


# Diagrama de bloques de un microcontrolador

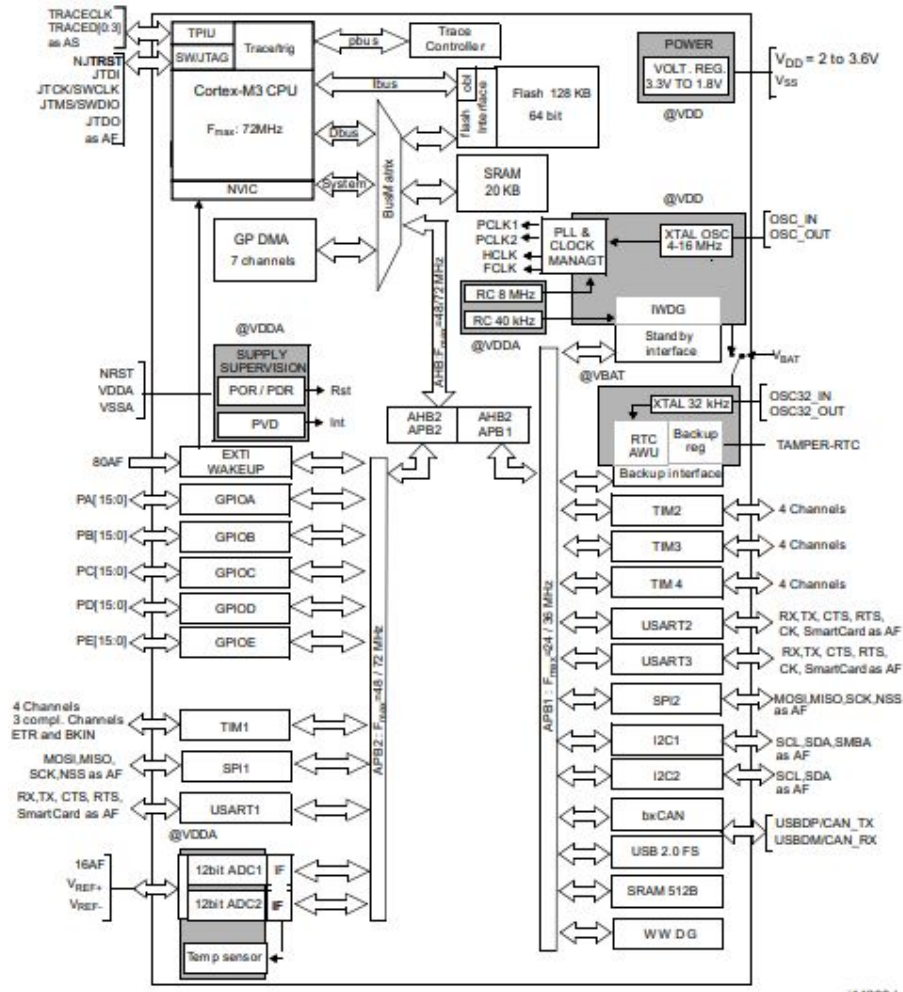




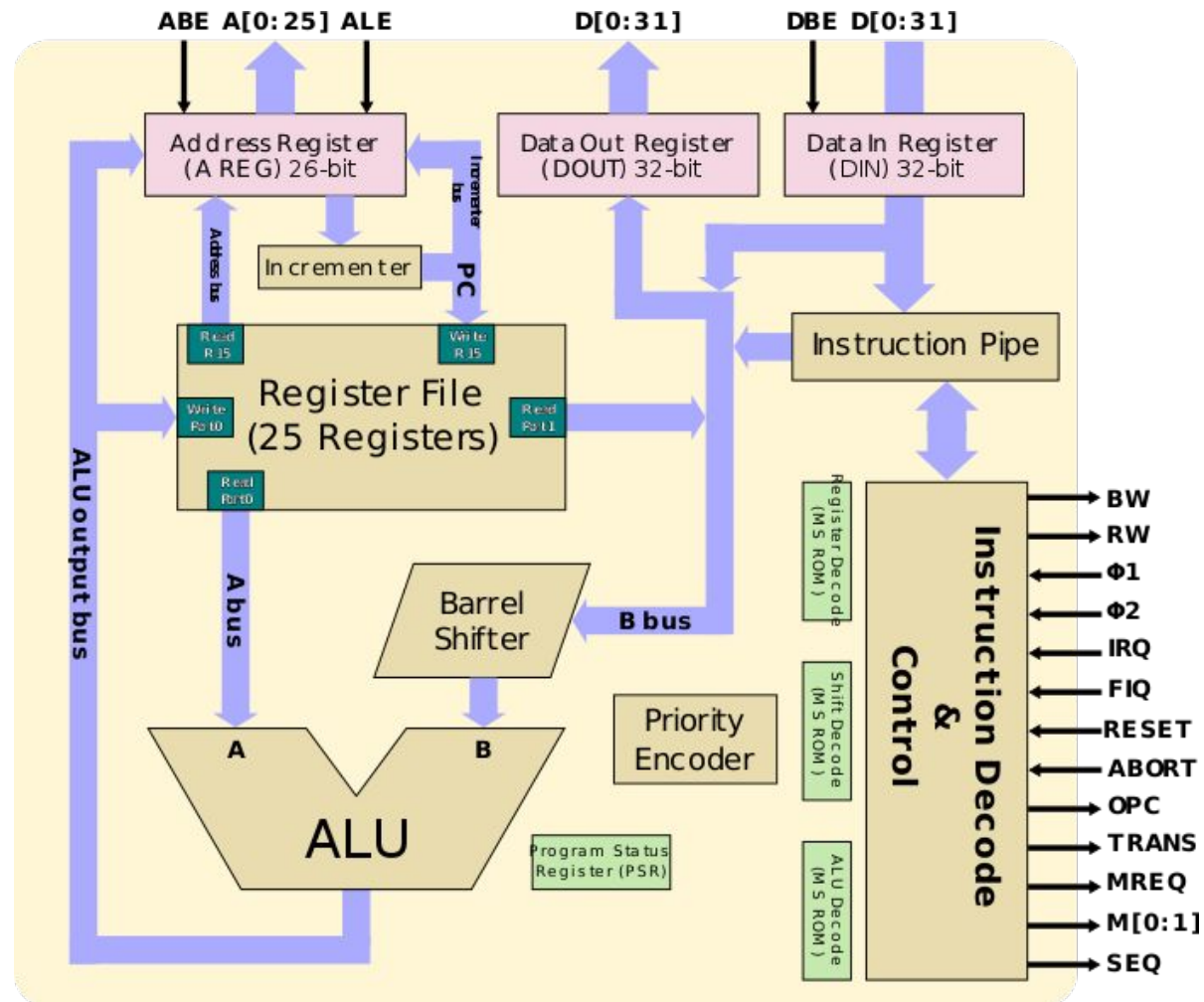
# Diagrama de bloques de un microcontrolador



# Diagrama de bloques de un microcontrolador



# Núcleo ARM



# Enmascaramiento y desplazamiento de bits

- Colocar el valor de 1 en ciertos bits
  - $R0 = R0 \mid 0b000001100;$
  - $R0 = R0 \mid 0b000000110;$
- Colocar el valor de 0 en ciertos Bits
  - $R1 = R1 \& 0b11110011;$
  - $R1 = R1 \& 0b11111110;$
- Desplazamientos de bits hacia la derecha y hacia la izquierda
  - $R1 = 7 \ll 2;$
  - $R2 = 2 \ll 4;$
  - $R3 = 48 \gg 2$

