

# SISTEMAS EMBEBIDOS I

## UNIDAD 1 – INTRODUCCIÓN A LOS SISTEMAS EMBEBIDOS

ELMER ALAN CORNEJO QUITO

# 1.1 QUÉ ES UN SISTEMA EMBEBIDO?

- Los sistemas embebidos (empotrados o Incrustados) son sistemas basados en microprocesadores o microcontroladores.
- Cumplen con tareas puntuales o de control.
- Casi todos los componentes necesarios para realizar esas tareas están incluidos en algún tipo de placa base.



# Algunos ejemplos de sistemas embebidos:



# Características de los sistemas

## embebidos

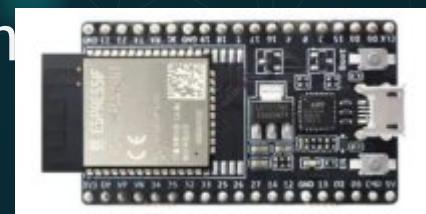
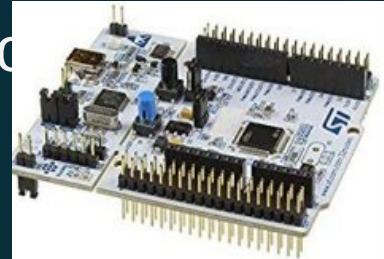
- Trabajan de forma autónoma, ininterrumpida y sin mantenimiento.
- Se modifican en función de las necesidades de uso.
- La industria es el sector que más se beneficia con esta tecnología.



- Existen plataformas open source para desarrollar aplicaciones de sistemas embebidos, como:

- Arduino.
- STM32.
- ESP8266 y ESP32.
- Raspberry Pi,
- etc.

- Muchos sistemas embebidos son sistemas de tiempo real.
- Pueden utilizar un sistema operativo real time (RTOS).



# 1.1 QUÉ ES UN SISTEMA EMBEBIDO?

## COMPONENTES DE LOS SISTEMAS EMBEBIDOS

- Microprocesador.
- Memoria RAM.
- Memoria Caché.
- Memoria No volátil.
- BIOS-ROM.
- CMOS-RAM.
- Reloj del Sistema.
- Chip Set.
- GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General).
- Módulos Internos: Temporizadores, ADC, DAC, UART, I2C, SPI, USB, BLUETOOTH,

# TIPOS DE SISTEMAS EMBEBIDOS

- SES: Sistemas embebidos pequeños
- LES: Sistemas Embebidos Grandes



# APLICACIONES DE LOS SISTEMAS

## EMBEBIDOS

- Los sistemas embebidos se emplean en ámbitos, como ser la militar, la electrónica de consumo, la salud, la automoción, la automatización, la domotización, las telecomunicaciones y, la industria.
- La industria es el sector que más está aprovechando la tecnología embebida.

Algunos beneficios en la industrial son:

- Control total.
- Conectividad y adaptabilidad.
- Reducción de costos.
- Diseño modular.
- Corto tiempo de respuesta.
- Accesibilidad.

## 1.2 COMPILADOR E INTERPRETE

El programa editado es llevado a la máquina que lo ejecutará finalmente, en este caso un microcontrolador comúnmente llamado “target”

- El programa es convertido a código de máquina al 100% y luego se ejecuta.
- El programa es enviado al “target” tal cual fue editado, y éste toma línea por línea y lo va convirtiendo a código de máquina a medida que va pasando por él.

## 1.2 COMPILADOR E INTERPRETE

Un **compilador** es un programa de software que transforma uno o varios archivos de código fuente y genera un archivo en código de máquina llamado ejecutable; este nuevo archivo es enviado al “target” para que lo ejecute.



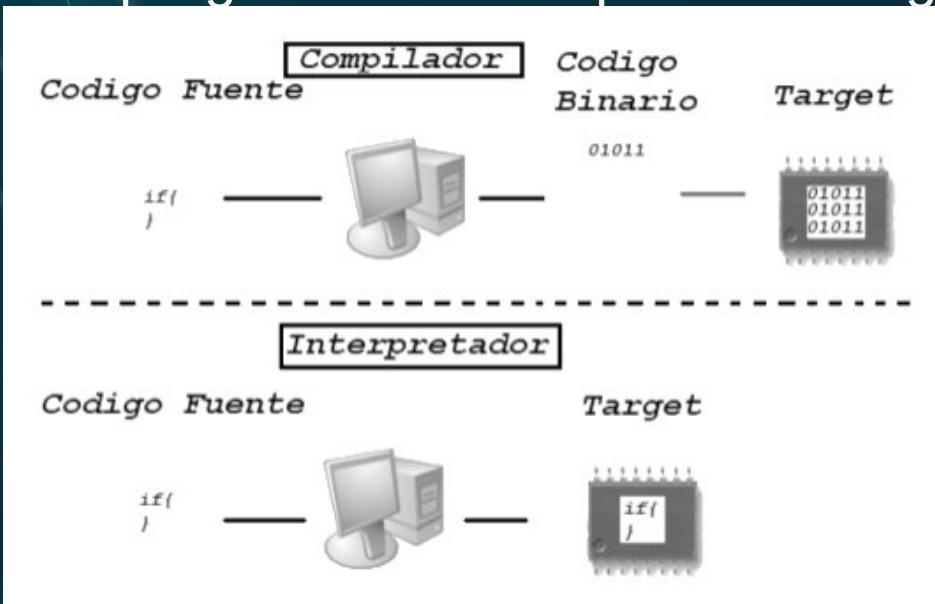
Un archivo compilado se ejecuta directamente en el microcontrolador porque está convertido a lenguaje máquina; en cambio, un archivo interpretado requiere un software incluido en el microcontrolador para su traducción.



# 1.2 COMPILADOR E

## INTERPRETE

Un **interpretador** es un software que es instalado en el “target”, el cual está preparado para recibir un archivo fuente editado; una vez el “target” recibe la orden de ejecutar inicia el proceso de cambiar línea por línea de programa a su respectivo código de máquina y ejecutar este



# ENTORNOS DE

## DESARROLLO

- Es una plataforma informática que facilita el desarrollo de código de programación.
- También se conoce como Integrated Development IDE.
- Combina herramientas que permiten automatizar fases de programación, como el análisis, diseño de arquitectura, codificación, pruebas, validación, gestión y mantenimiento.
- Normalmente, consiste en un editor de código fuente, herramientas de construcción automáticas, depurador.
- Generalmente, el entorno de desarrollo trabaja con tres niveles de servidores: desarrollo, montaje o

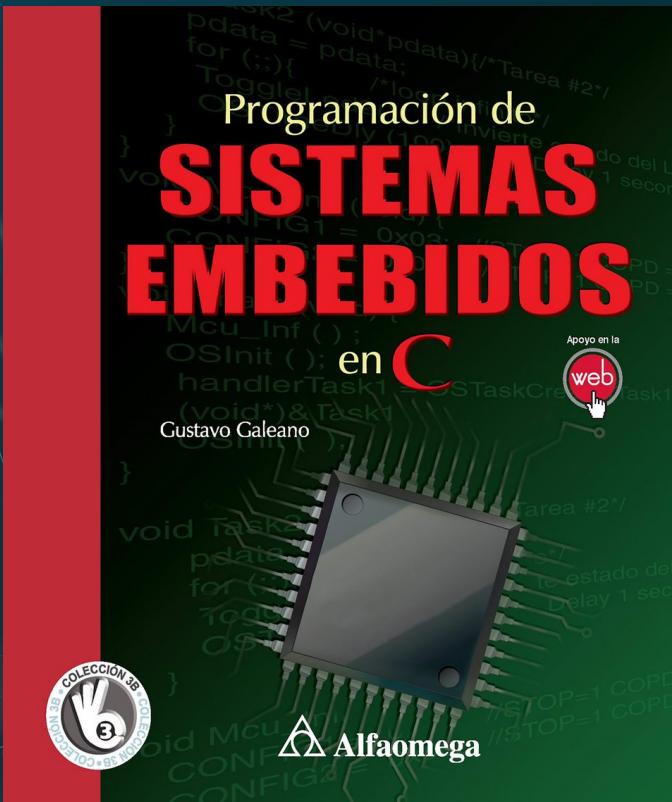


# CONCLUSIONES

Después de todo lo que hemos visto hasta aquí, es evidente que los sistemas embebidos son parte de nuestra vida diaria y que las posibilidades que ofrecen para todo sector son casi ilimitadas. El desarrollo de software empotrado es algo tan común actualmente que incluso lenguajes de programación no tradicionales del ámbito, como por ejemplo Python y JavaScript, se utilizan en entornos embebidos.

# BIBLIOGRAFIA

Galeano, G. (2009). *Programación de Sistemas Embebidos en C*. México: Alfaomega.



# 1.3 ARQUITECTURA DE

## MICROCONTROLADORES (MCU(microcontroller unit))

- Fundamentos de microcontroladores (MCUs)
  - Son computadores digitales integrados en un chip que cuentan con un microprocesador o unidad de procesamiento central (CPU), una memoria para almacenar el programa, una memoria para almacenar datos y puertos de entrada salida. (Torres M. Tutorial Microcontroladores PIC)
  - Destinada a realizar una tarea específica comandada por el programa que existe cargado en el chip.
  - Contiene todo lo necesario para interactuar con el usuario y llevar a cabo una tarea en específico. Todo encapsulado en un solo chip.



# 1.3 ARQUITECTURA DE MICROCONTROLADORES

- Fundamentos de microcontroladores (MCUs)

- Las aplicaciones de los microcontroladores:

- Electrodomésticos (microondas, lavadoras, máquinas de café,...).

- Telecomunicaciones (teléfonos móviles).

- Industria automotriz (inyección de combustible, el tablero de control,...).

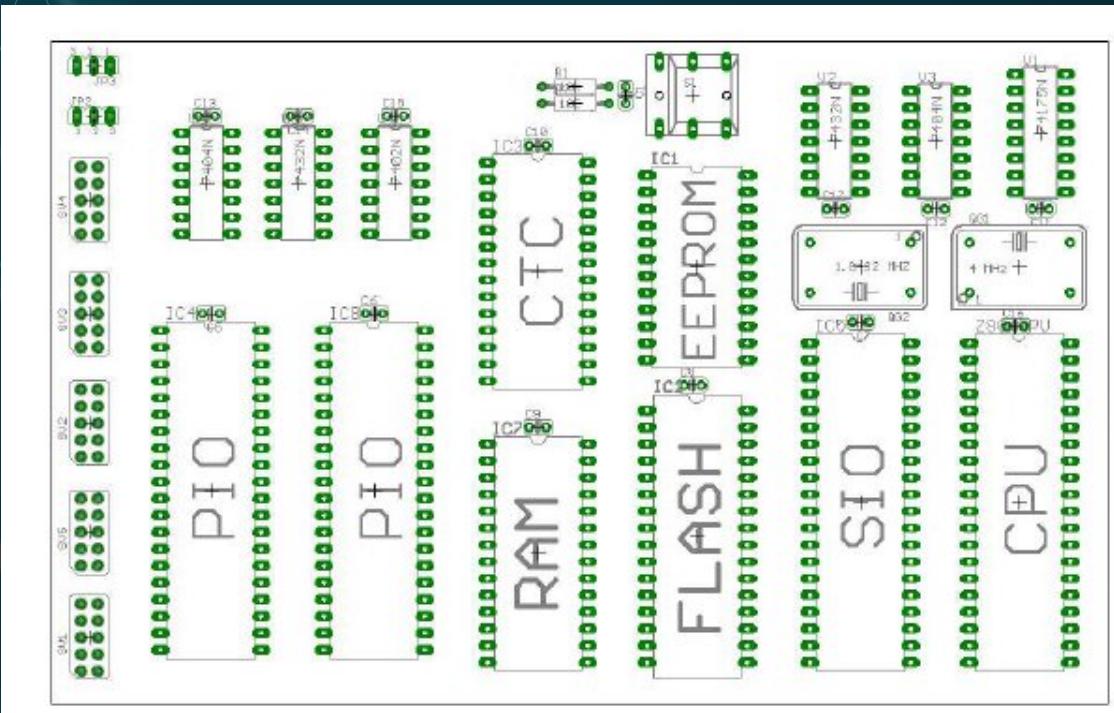
- Industria aeroespacial.

- Automatización industrial.

# 1.3 ARQUITECTURA DE

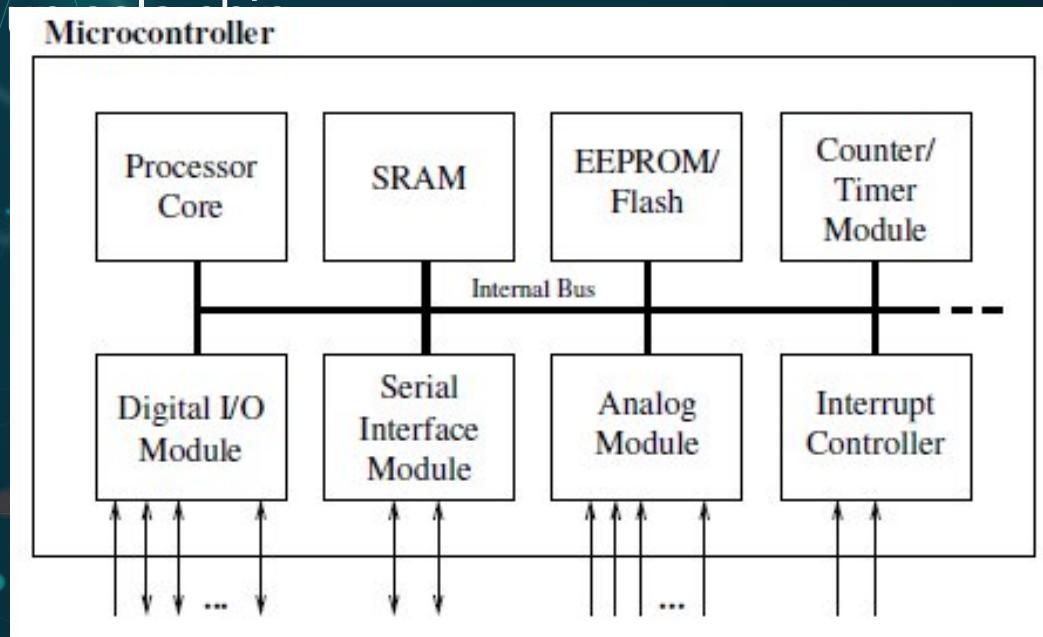
# Fundamentos de microcontroladores (MCUs)

El circuito se vería así:



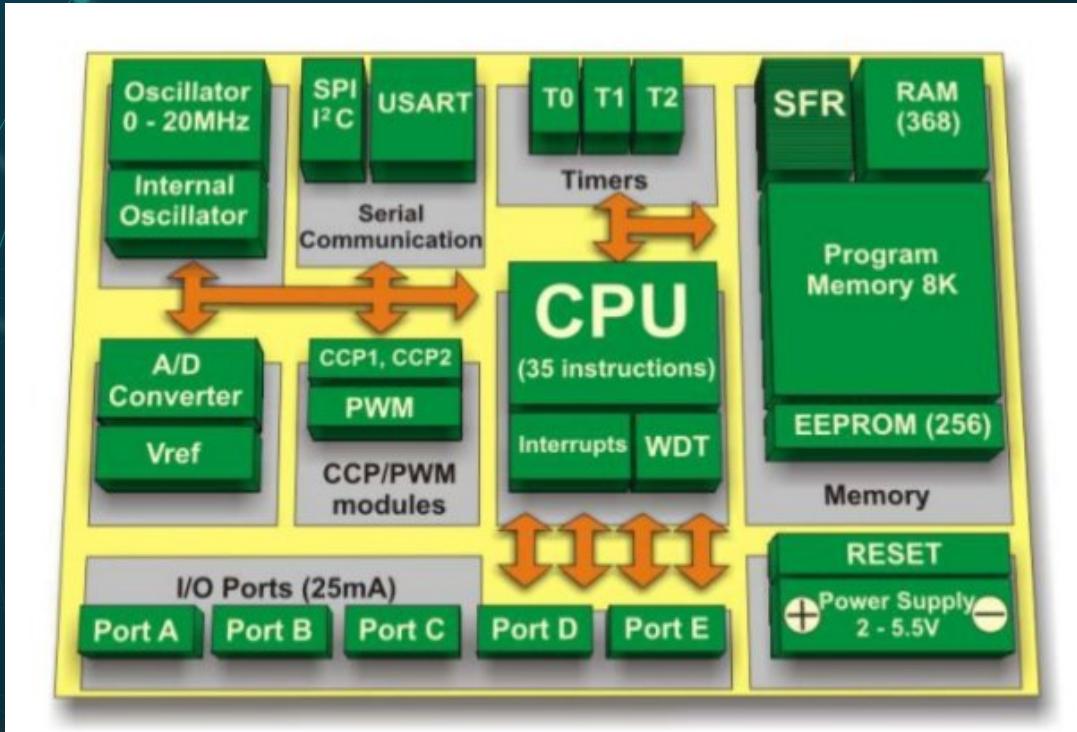
# 1.3 ARQUITECTURA DE MICROCONTROLADORES

Se demuestra con este ejemplo las diferencias entre un microcontrolador y un microprocesador. Un microcontrolador es un microprocesador con memorias, periféricos de entrada y salida y entre otros componentes, integrados en



# 1.3 ARQUITECTURA DE MICROCONTROLADORES

Un microcontrolador viene con todo lo necesario, encapsulado en un solo chip, para trabajar sobre una aplicación en específico.

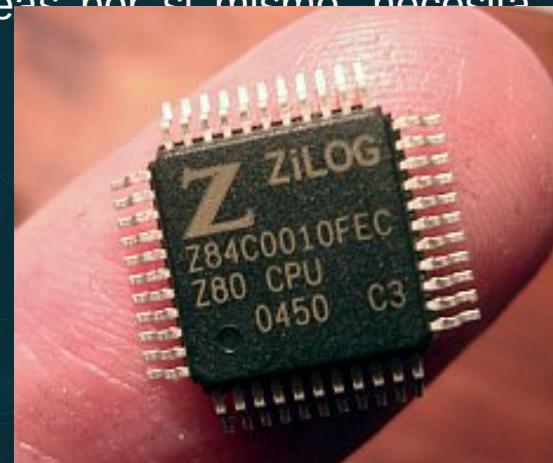


## 1.3 ARQUITECTURA DE

Viene a ser prácticamente la Unidad Central de Procesamiento (CPU, Central Processing Unit).

### MICROCONTROLADORES

1. Consiste principalmente en la ALU (Arithmetic Logic Unit), un codificador de instrucciones y los registros.
2. Se constituye en el cerebro del microcontrolador.
3. El microprocesador NO puede realizar todas las tareas por si mismo, necesita de circuitos de apoyo como:
  4. • Microprocesador.
  5. • Memoria RAM.
  6. • Memoria Caché.
  7. • Memoria No volátil.
  8. • BIOS-ROM.
  9. • CMOS-RAM.
  10. • Reloj del Sistema.
  11. • Chip Set.
  12. • GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General).
  13. • Módulos Internos: Temporizadores, ADC, DAC, UART, I2C, SPI, USB, BLUETOOTH, WIFI, etc.

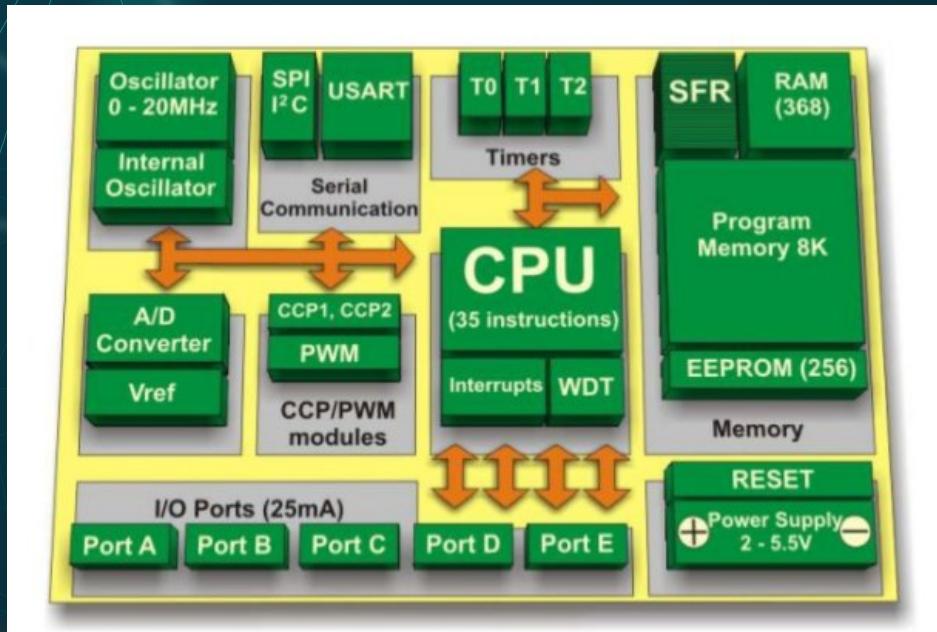


# 1.3 ARQUITECTURA DE MICROCONTROLADORES

- CPU : Contiene la unidad de lógica aritmética, la unidad de control y los registros (puntero de pila, contador de programa, registro de acumulador, archivo de registro, ...).
- Memoria: Se divide en memoria de programa y memoria de datos. En los controladores más grandes, Un controlador DMA controla las transferencias de datos entre componentes periféricos y la memoria.
- Controlador de interrupción: Las interrupciones son útiles para interrumpir el flujo normal del programa en caso de eventos externos o internos importantes
- Temporizador / Contador: La mayoría tienen por lo menos uno (2-3 temporizadores / contadores). Usados para marcar eventos de tiempo, medir intervalos o contar eventos.
- Muchos controladores también contienen salidas PWM (modulación de ancho de pulso), que pueden usarse para diversos casos
- E/S digitales: Los puertos de E / S digitales paralelos son una de las principales características de los microcontroladores. El número de los pines de E / S varía de 3-4 a más de 20, dependiendo de la familia de controladores y del

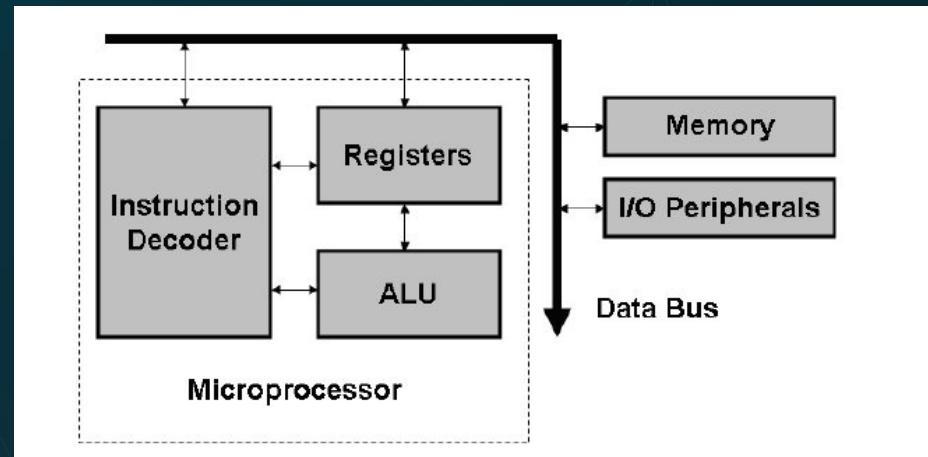
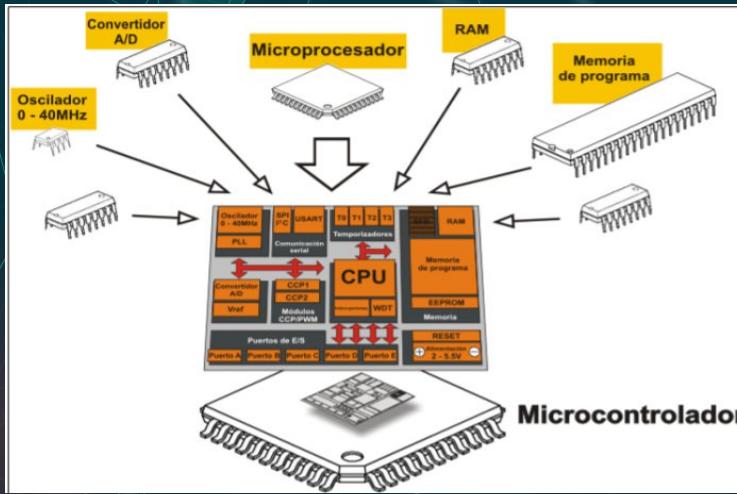
# 1.3 ARQUITECTURA DE MICROCONTROLADORES

**Watchdog Timer:** Se utiliza para protegerse contra errores en el programa y/o el hardware. El temporizador de vigilancia se utiliza para reiniciar el controlador en caso de bloqueo del software.



# 1.3 ARQUITECTURA DE MICROCONTROLADORES

- Microprocesador vs Microcontrolador
  1. Un microcontrolador se utilizan para funciones específicas mientras que los microprocesadores se utilizan para tareas más generales.
  2. El microprocesador, necesita de periféricos para poder funcionar, memorias ROM, RAM, etc. El microcontrolador ya tiene todo incluido en un solo chip.

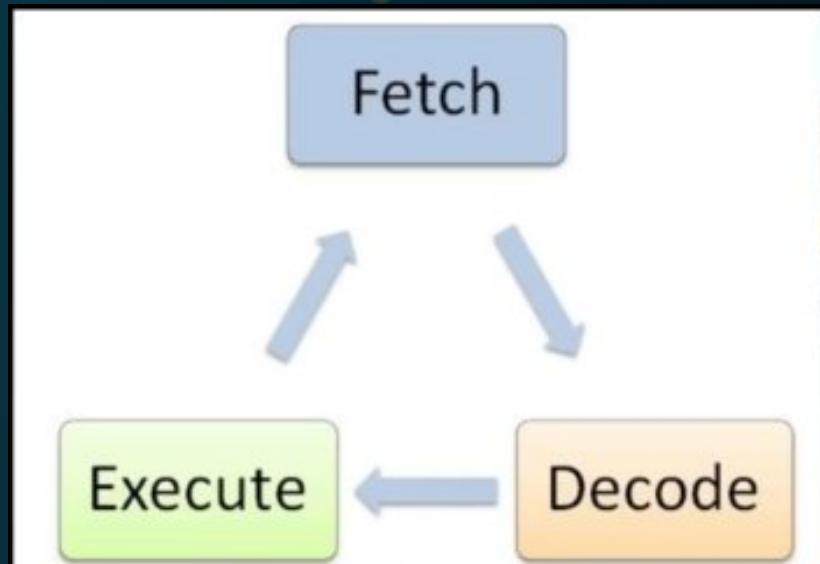
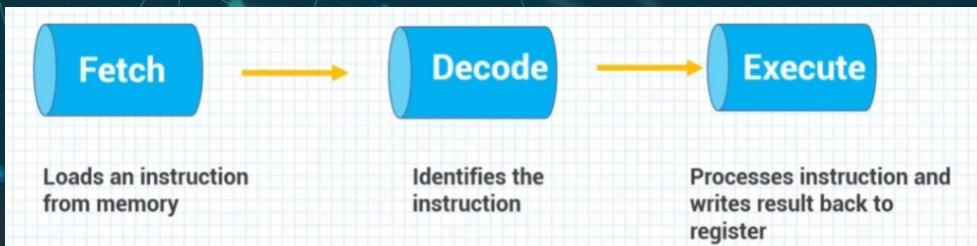


# 1.4 PIPELINE

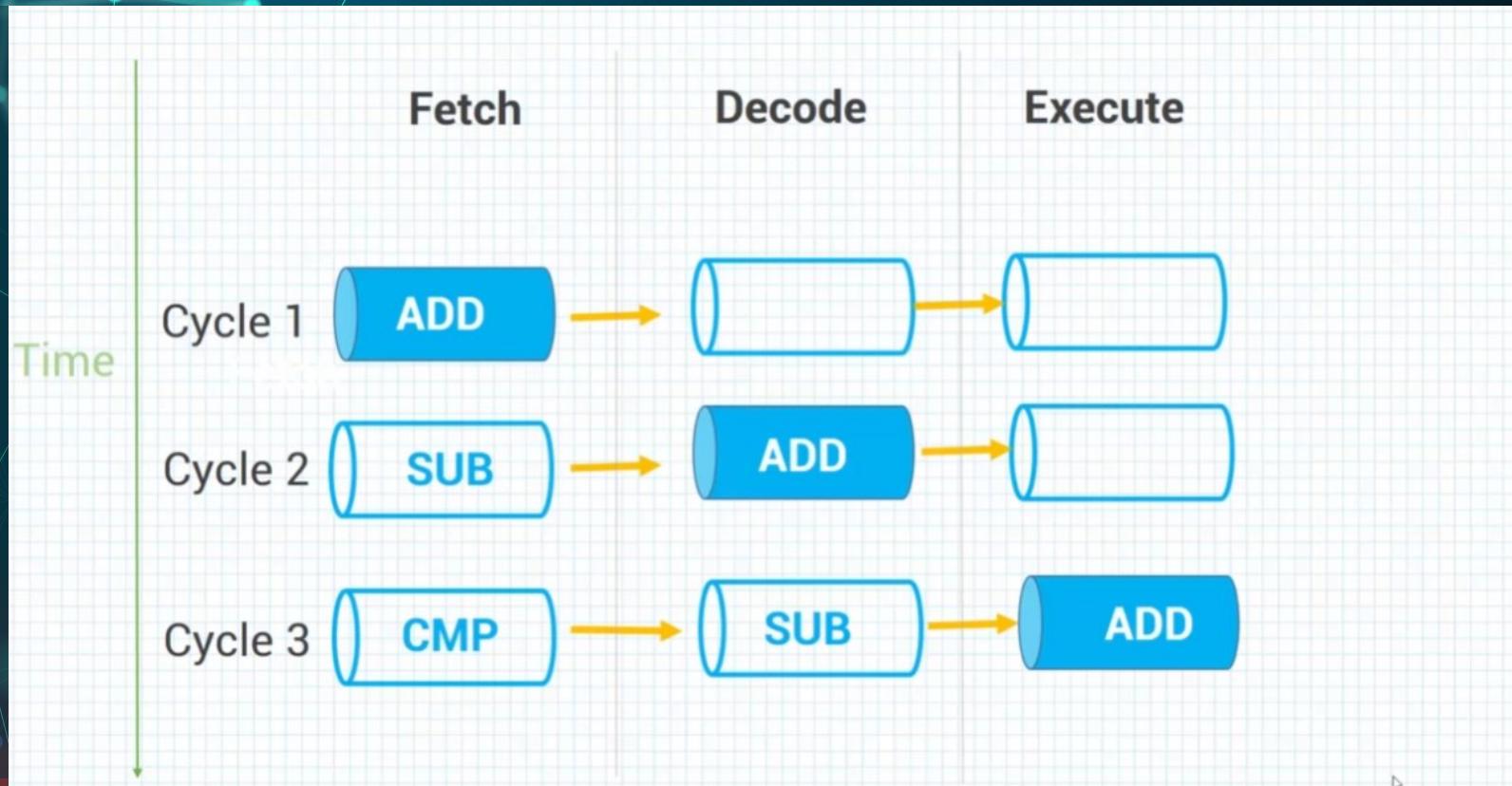
Fetch: Búsqueda

Decode: Decodificación

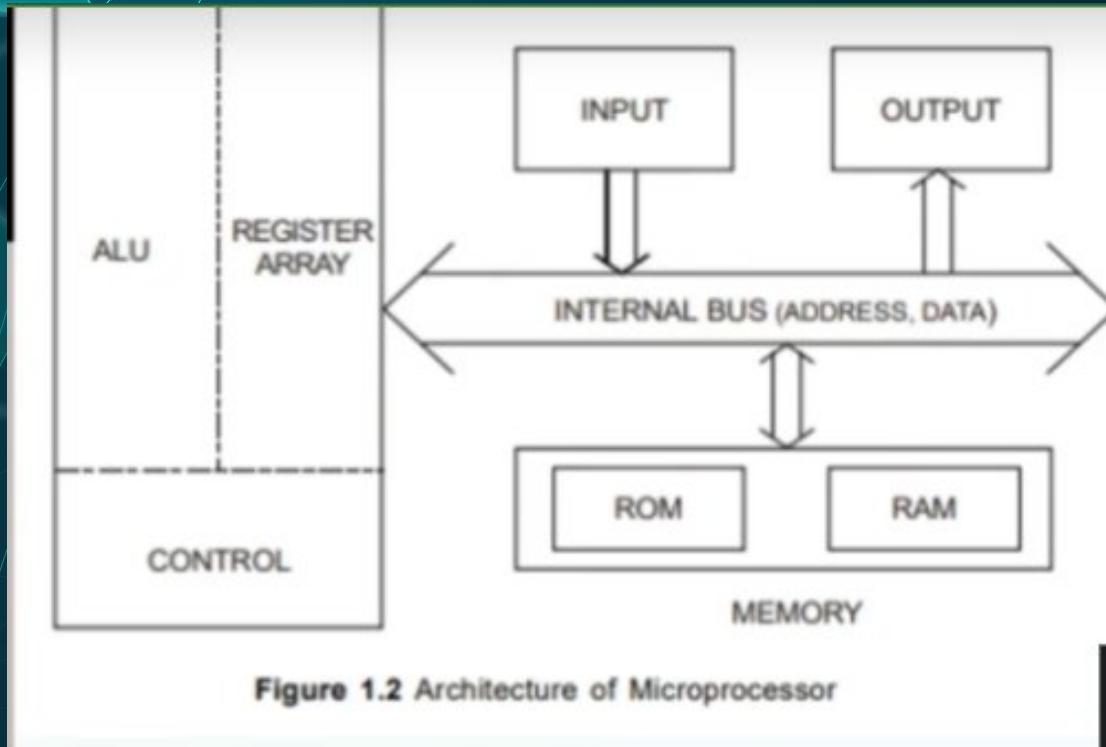
Execute: Ejecución



# 1.4 PIPELINE

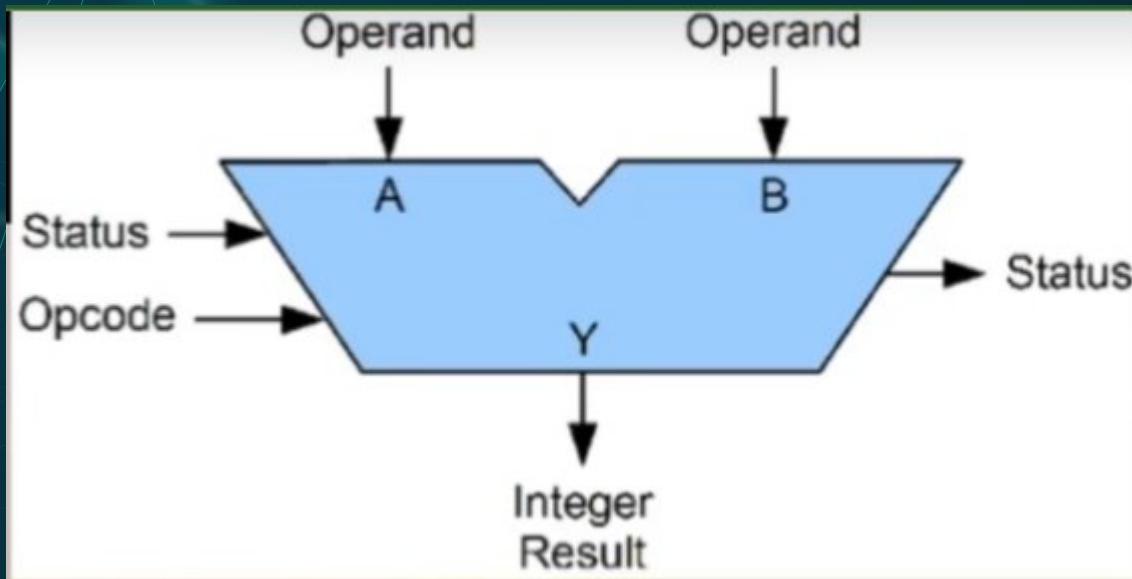


## 1.4 PIPELINE



## 1.4 PIPELINE

ALU (arithmetic logic unit) UNIDAD LOGICA ARITMETICA



## 1.5 CLASIFICACION DE ARQUITECTURAS

Unidad de Control. El CPU está constantemente ejecutando instrucciones de programa. Es la función de la Unidad de Control determinar que operaciones deben ser ejecutadas y configuradas para que el “data path” las realice.

El registro “program counter (PC)” es utilizado para almacenar las direcciones de la siguiente instrucción de programa.

La configuración del “data path” incluye proveer las entradas apropiadas para la ALU, seleccionar la operación correcta de la ALU y asegurar que la dirección está siendo escrita en el lugar correcto.

## 1.5 CLASIFICACION DE ARQUITECTURAS

- Unidad de control: Tradicionalmente este componente era “hard-wired” que básicamente consistía en una tabla que contiene los valores de las líneas de control necesarias para realizar una instrucción en específico. Además de incluir una lógica de decodificación compleja.
  1. Esto dificultaba el hecho de extender el set de instrucciones del CPU.
  2. Como solución surge la propuesta de Maurice Wilkes, de añadir microinstrucciones a la Unidad de control para que se facilite esta tarea y como resultado nace: CISC (Complex Instruction Set Computer), que como desventaja trae consigo reducción en la velocidad.
  3. Con el objeto de mantener la velocidad se vuelve a la arquitectura “hard-wired” y se desarrolla la RISC (Reduced Instruction Set Computer).

## 1.5 CLASIFICACION DE ARQUITECTURAS

CISC. Arquitectura caracterizada por su complejo set de microinstrucciones que usualmente toman varios ciclos de reloj en realizarse. Se ofrece un set de instrucciones complejo y capaz de realizar acciones de mayor dificultad.

RISC. Tiene instrucciones simples “hard-wired” que normalmente toman uno o pocos ciclos de reloj en realizarse. Como resultado las instrucciones se llevan a cabo rápido pero su set de instrucciones son bastante simples.

## **CISC**

Computadoras con un conjunto complejo de instrucciones

- Instrucciones complejas multi-ciclo
- Instrucciones complejas (más de 1000)
- El código final es pequeño
- Muchos ciclos de reloj por segundo
- Alto costo de producción

## **RISC**

Computadoras con un conjunto reducido de instrucciones

- Instrucciones se pueden ejecutar en un solo ciclo de reloj
- Instrucciones sencillas (menos de 100)
- El tamaño final del código es grande
- Pocos ciclos de reloj por segundo
- Bajo costo de producción

## 1.5 CLASIFICACION DE ARQUITECTURAS

Von Neumann Architecture. Cuenta con el programa y los datos almacenados en uno solo. Pueden ser accedidos por el mismo bus de datos.

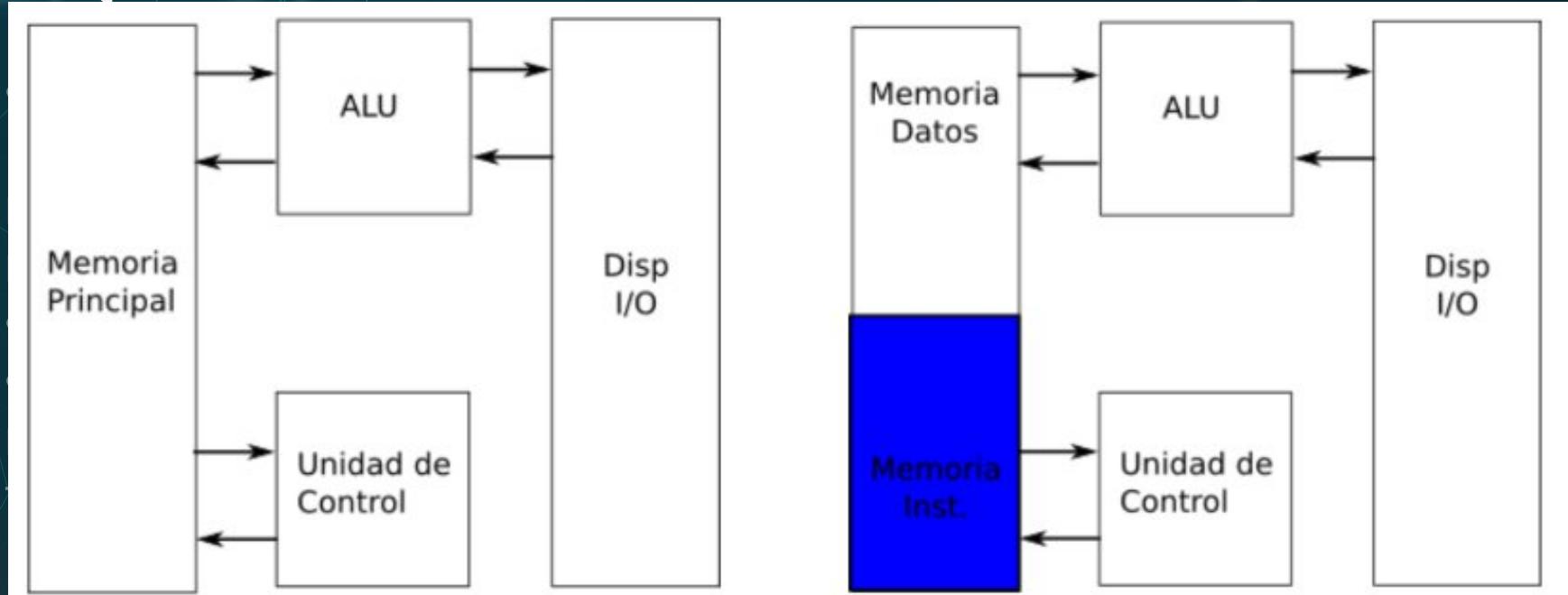
De este hecho, algunos problemas pueden darse y causar retardos inesperados.

Harvard Architecture. El programa y los datos están en memorias separadas. Accedidas por buses distintos, de este modo se tiene un desempeño del sistema óptimo.

Como una desventaja requiere de más hardware, de dos buses de datos, dos chips de memoria, etc.

# 1.5 CLASIFICACION DE ARQUITECTURAS

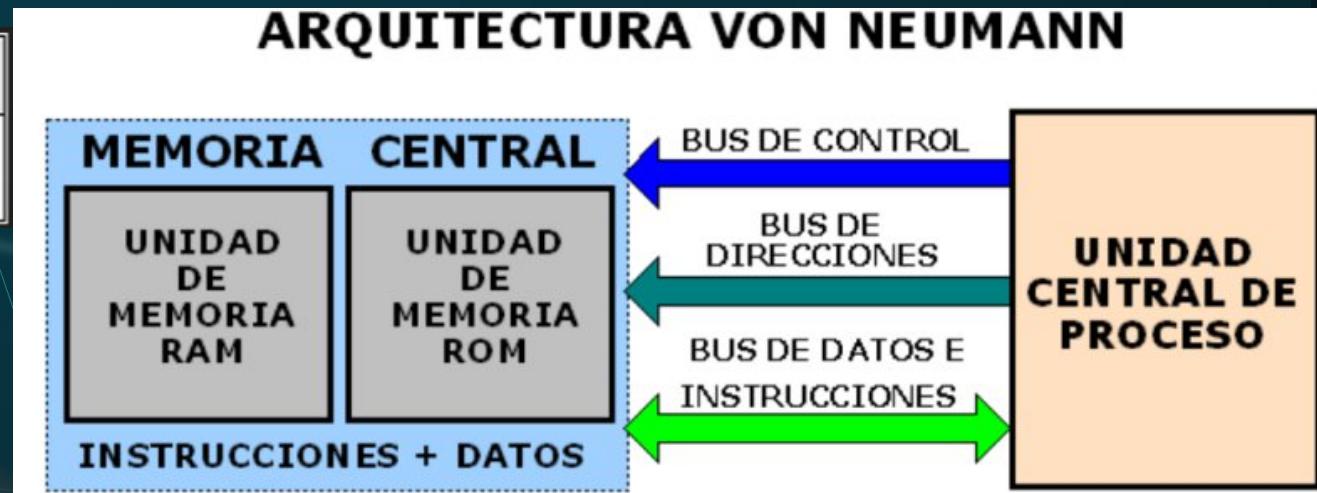
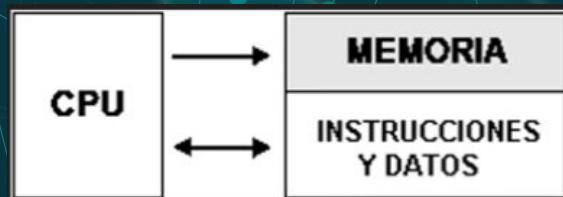
## ARQUITECTURA VON NEUMANN Y HARVARD

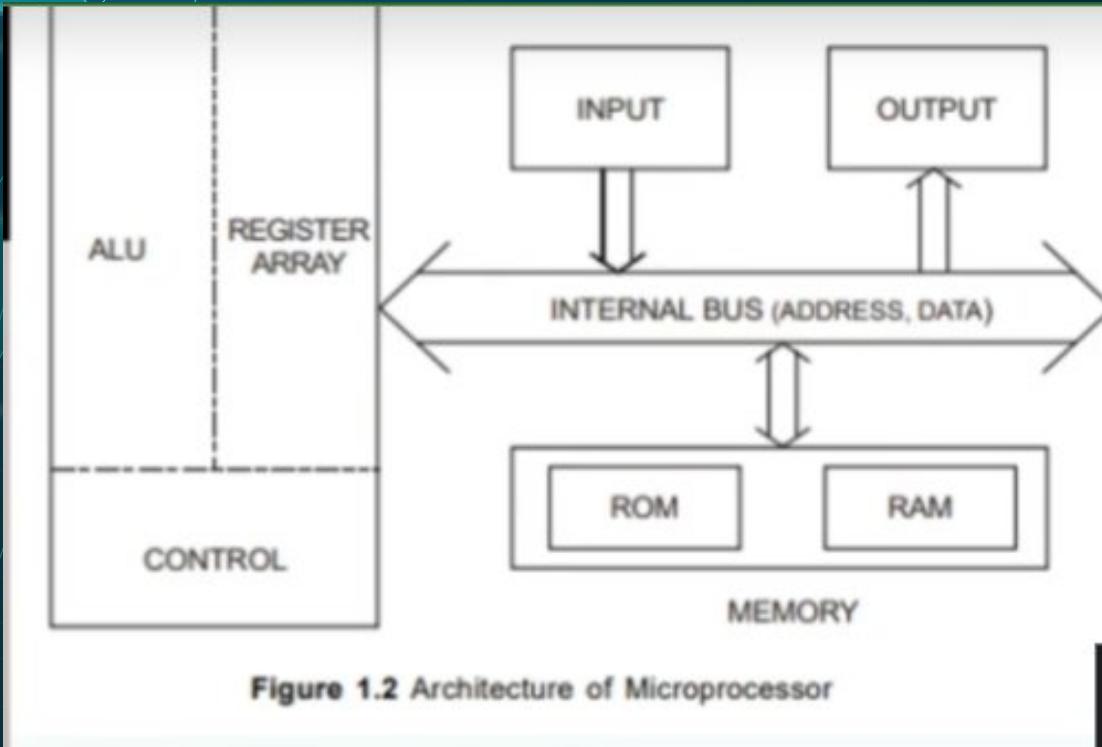


# 1.5 CLASIFICACION DE ARQUITECTURAS

ARQUITECTURA VON NEUMANN

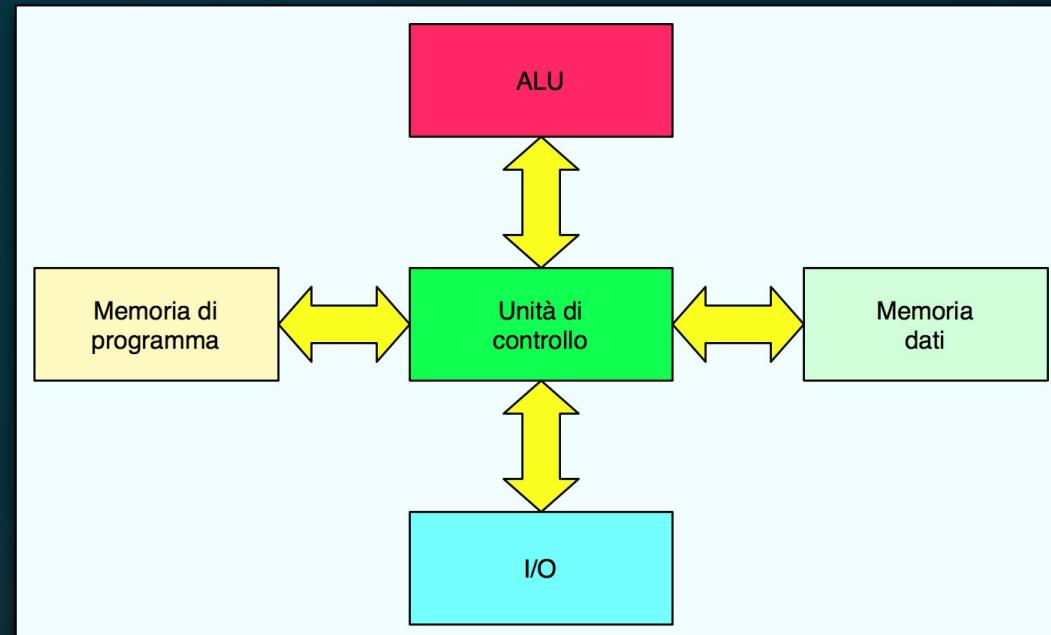
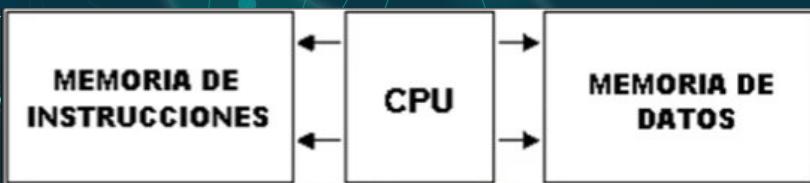
Una de sus características es que la memoria comparte los datos y el código





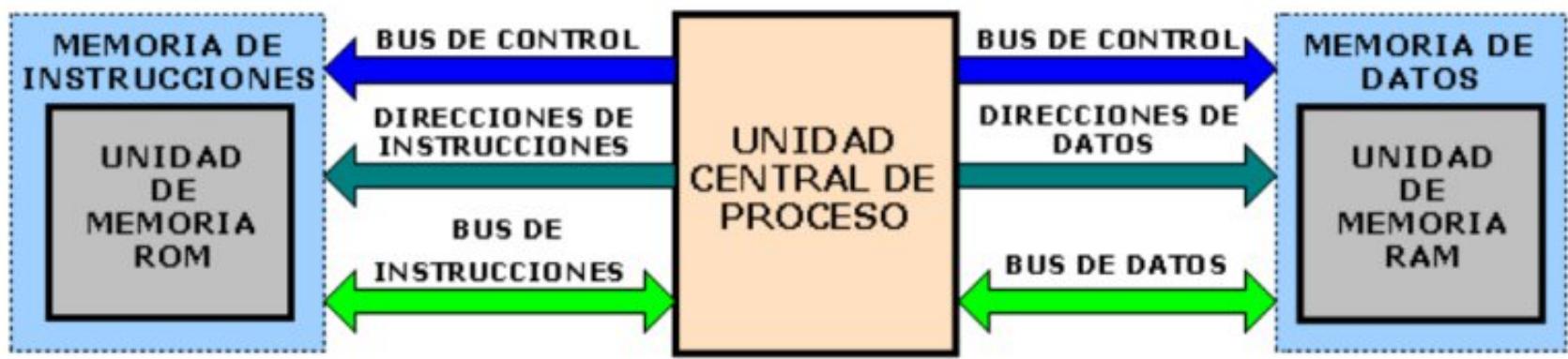
# 1.5 CLASIFICACION DE ARQUITECTURAS

ARQUITECTURA HARVARD  
Una de sus características es que tiene una memoria específica para el código y otra memoria específica para los datos.



# 1.5 CLASIFICACION DE ARQUITECTURAS

## ARQUITECTURA HARVARD



# 1.5 CLASIFICACION DE ARQUITECTURAS

## RISC

Es una arquitectura con conjunto de instrucciones reducido

Hace hincapié en el software para optimizar el conjunto de instrucciones

Es una unidad de programación cableada en el procesador

Requiere múltiples conjuntos de registros para almacenar la instrucción

## CISC

Es una arquitectura de conjunto de instrucciones complejas

Hace hincapié en el hardware para optimizar el conjunto de instrucciones

Unidad de microprogramación en el procesador

Requiere un único conjunto de registros para almacenar la instrucción

# 1.5 CLASIFICACION DE ARQUITECTURAS

## CISC

Tiene una sencilla decodificación de las instrucciones

Los usos de la canalización son simples

Utiliza un número limitado de instrucciones que requiere menos tiempo para ejecutar las instrucciones

Utiliza LOAD y STORE que son instrucciones independientes en el registro para registrar la interacción de un programa

## RISC

Tiene una compleja decodificación de las instrucciones

Los usos de la canalización son difíciles

Utiliza una gran cantidad de instrucciones que requieren más tiempo para ejecutar las instrucciones

Utiliza la instrucción LOAD y STORE en la interacción de memoria a memoria de un programa

# 1.5 CLASIFICACION DE ARQUITECTURAS

## CISC

Tiene más transistores en registros de memoria

El tiempo de ejecución es muy corto

Se puede utilizar con aplicaciones de gama baja como domótica, sistema de seguridad, etc.

Tiene instrucciones de formato fijo

El programa escrito para esta arquitectura necesita ocupar más espacio en la memoria

## RISC

Tiene transistores para almacenar instrucciones complejas

El tiempo de ejecución es mayor

Se puede utilizar con aplicaciones de alta gama como telecomunicaciones, procesamiento de imágenes, procesamiento de video, etc.

Tiene instrucciones de formato variable

Los programas escritos para esta arquitectura tienden a ocupar menos espacio en la memoria

# 1.5 CLASIFICACION DE ARQUITECTURAS

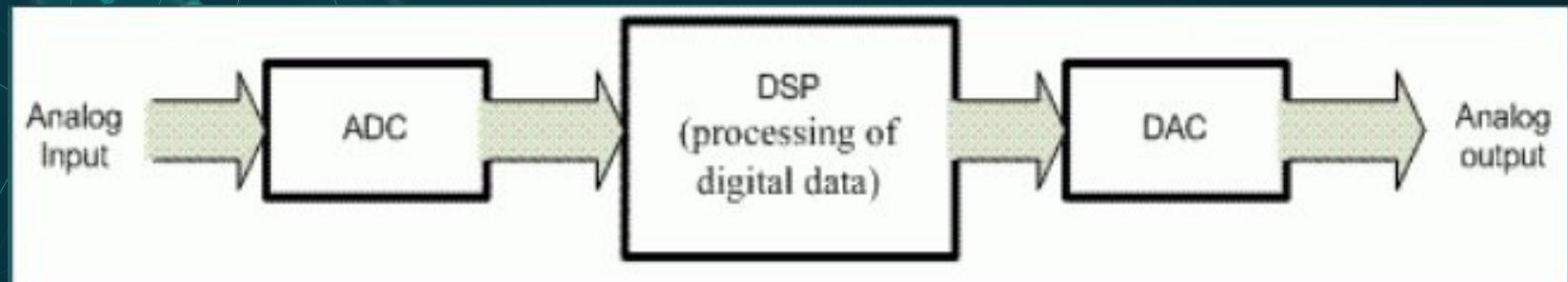
- Arquitectura Von Neumann – memoria común, bus de datos e instrucción.
- Arquitectura Harvard – memorias separadas, buses de datos e instrucciones
- Arquitectura Super Harvard – memorias separadas + CACHE para instrucciones + controladores I/O (E/S).
- La arquitectura Harvard es típica para microcontroladores y DSP.

## 1.5 CLASIFICACION DE

### ARQUITECTURAS

DSP – Procesador Digital de Señal, es un microprocesador especializado, diseñado para el procesamiento digital de señales, con mayor frecuencia

en



# 1.5 CLASIFICACION DE ARQUITECTURAS

- ARM - abreviación de inglés Advanced RISC Machine (Máquina Avanzada RISC) – una de las más ampliamente utilizadas microprocesadores de 16/32-bit y microcontroladores en el mundo de los dispositivos de telefonía móvil.
- El desarrollo original de los núcleos ARM pertenece a la compañía Acorn Computers Ltd., pero actualmente los microprocesadores y microcontroladores basados en ARM son producidos por muchas compañías: Alcatel, Atmel, NEC, NVIDIA, NXP (anteriormente Philips), Oki, Qualcomm, Samsung, Sharp, ST Microelectronics, Symbios Logic, Texas Instruments, VLSI Technology etc.
- Desde 2009, los procesadores ARM representan casi el 90% de todos los procesadores RISC embebidos de 32-bit (electrónica de consumo, asistentes digitales personales - PDA, teléfonos móviles, reproductores iPod y otros medios de comunicación digitales y reproductores de audio, consolas de juegos, calculadoras, HDD, routers etc.).
- Familia ARM MP incluye ARM7, ARM9, ARM11 y Cortex. Las velocidades de reloj varían ampliamente – de decenas de MHz a 1GHz.
- ARMs son una muy buena combinación de la lógica moderna, de alta funcionalidad, bajo consumo (en su mayoría son CMOS), de bajo coste, la arquitectura es simplificada, lo que permite una fácil integración en varios dispositivos.

# 1.5 CLASIFICACION DE ARQUITECTURAS

Sistemas embebidos, que utilizan ARM

- Automotive: Automotor
- Microcontrollers: Microcontrolador
- Wireless: sin cable
- Imaging: escaneo
- Storage: Almacén
- Networking: la creación de redes
- Security: seguridad
- Consumer: consumido



# 1.5 CLASIFICACION DE ARQUITECTURAS

- Intel y Motorola basicamente utilizan arquitectura von Neumann con memoria común, bus de datos e instrucciones
- Familias PIC MC de Microchip – RISC MC con arquitectura Harvard
  - Familia PIC12CXXX/PIC12FXXX 8-pin instrucciones 12/14-bit
  - Familia PIC16C5X instrucciones de 12-bit
  - Familia PIC16CXXX/PIC16FXXX instrucciones de 14-bit
  - Familia PIC17CXXX instrucciones de 16-bit
  - Familia PIC18CXXX/PIC18FXXX modelo avanzado, instrucciones de 16-bit
  - Familia PIC32MX3xx 32-bit MC con velocidades de reloj más altas y la memoria y muchos otros dispositivos embebidos.

# 1.5 CLASIFICACION DE ARQUITECTURAS

## ATMEGA328

Microcontrolador

Es un CMOS de 8 bits de bajo consumo, basado en AVR con arquitectura RISC (Reduced Instruction Set Computing).

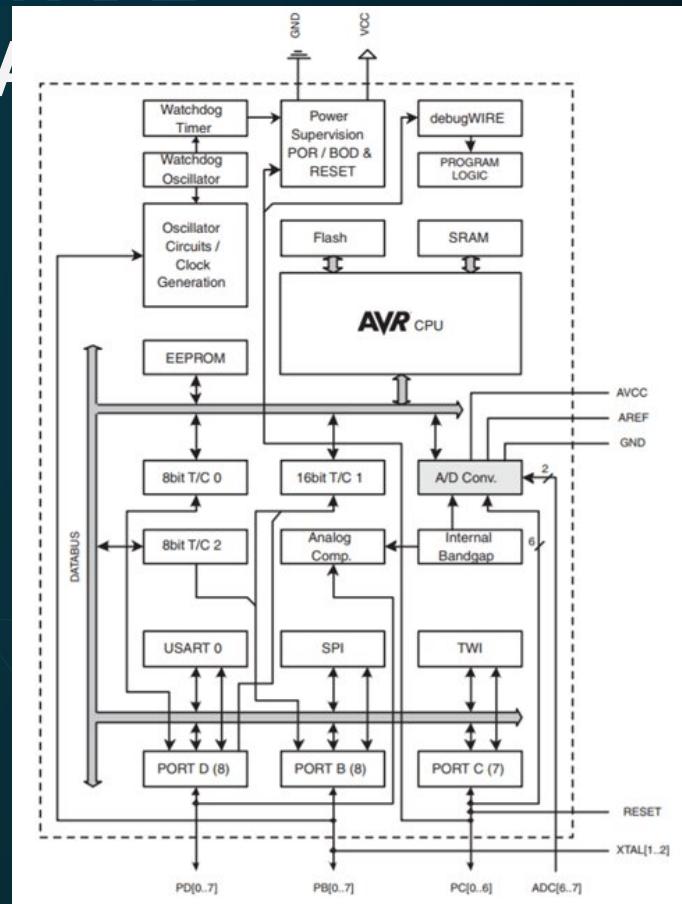
8 bits hace referencia al tamaño de los buses de datos, externos e internos. Asimismo al tamaño de los registros internos que se tienen.

Tendrá también puertos E/S de 8 bits.

Alcanza una velocidad aproximada de 1 MIPS por MHz.

MIPS es Million Instructions Per Second

Tiene 23 líneas de E/S de propósito general, 32 registros de propósito general.



# 1.5 CLASIFICACION DE ARQUITECTURA

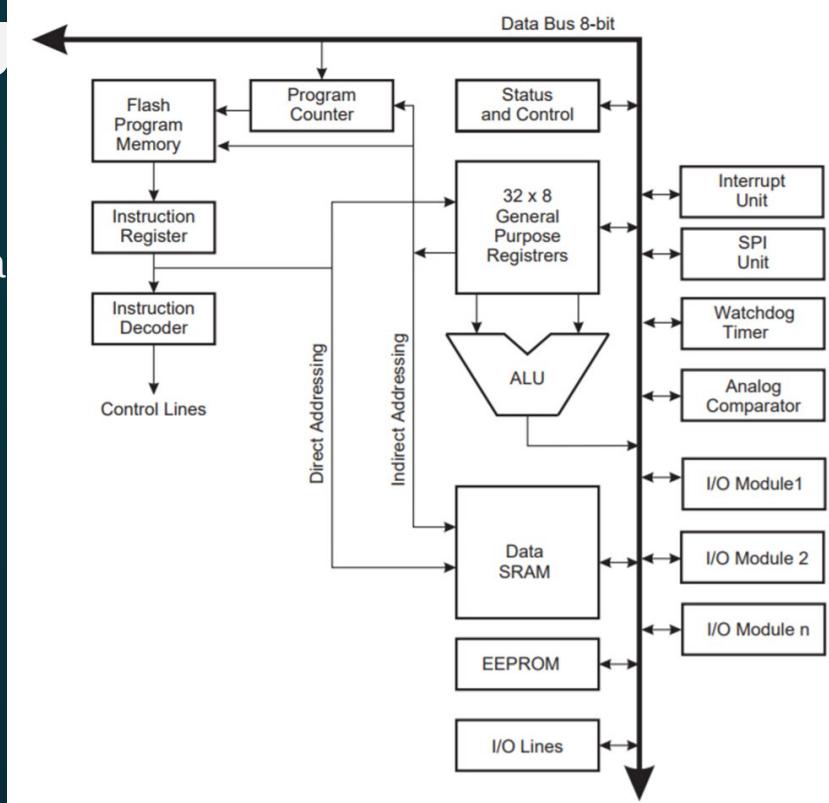
ATMEGA328

Microprocesador

Se basa en un AVR CPU.

AVR utiliza la arquitectura Harvard, cuya característica principal es que mientras una instrucción se ejecuta la siguiente es pre-extraída de la memoria.

Se alcanza velocidades aproximadas de 1 MIPS (Millions Instructions Per Second) por MHz.

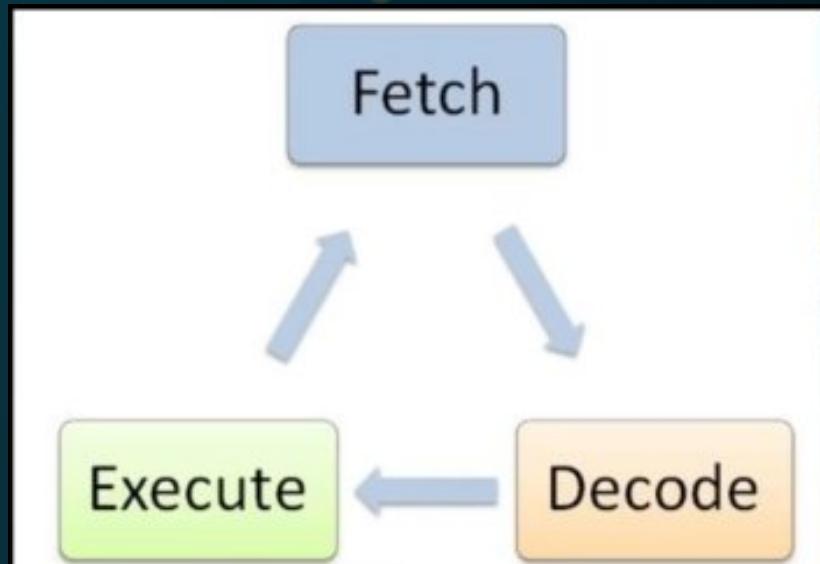
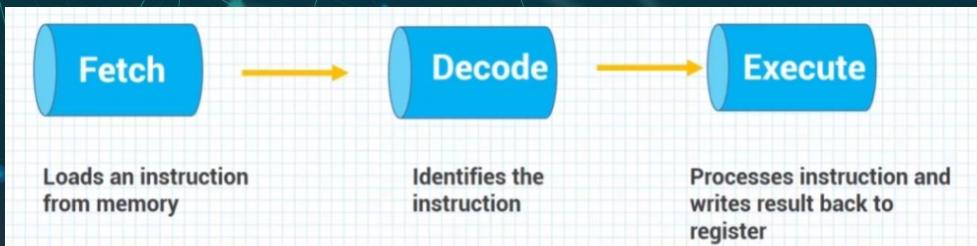


# 1.4 PIPELINE

Fetch: Búsqueda

Decode: Decodificación

Execute: Ejecución



# 1.5 CLASIFICACION DE ARQUITECTURAS

## Buses

A los Buses se les denomina también sistemas de líneas para la conexión interna y externa entre los dispositivos en un Sistema informático.

Dependiendo de los dispositivos que se conectan, se pueden distinguir: un bus de sistema (bus principal), buses internos para la conexión con la memoria RAM principal, la conexión con la memoria Caché, buses de entrada/salida I/O, etc.

Principales tipos de buses:

1. Bus de direcciones
2. Bus de datos
3. Bus de control

## Bus de direcciones

- La dirección es un número binario, identificando un lugar definido de la memoria o un Puerto de E/S, que participa en la transferencia de datos.
- El bus de direcciones está diseñado para enviar las direcciones, preparadas en el microprocesador, con el objetivo de elegir una celda definida de la memoria o un Puerto I/O (Entrada/salida)
- El bus de direcciones es de un solo sentido: las direcciones siempre son generadas por la MS.
- El ancho del bus de direcciones: determina el tamaño de la memoria, que puede ser direccionado directamente por el microprocesador.

## Bus de datos

■ A lo largo del bus de datos de intercambio de información (instrucciones o datos) se lleva a cabo entre el microprocesador y los dispositivos periféricos – se trata de un intercambio de dos vías.

Ejemplos de transferencia de datos:

1. La lectura de las instrucciones de programación de la memoria.
2. El envío de datos desde el Sistema de MP a los puertos de E/S (I/O)
3. La lectura de datos desde los puertos de E/S y enviarlos al Sistema MP.
4. El envío de los resultados del Sistema de MP a la memoria.

Se trata de operaciones de lectura y escritura.

Ancho de banda del bus de datos: 8086: 16 bits 80486: 32 bits, Pentium:

## Bus de Control

- El bus de control es utilizado para el envío y la recepción de señales de control.
- Las señales de control aseguran la sincronización (control del tiempo) entre el MS y el resto de los componentes del Sistema:
- Típicas señales de control:
  - RD (lectura) y WR (escritura) – señales de control sobre lectura y escritura
  - reloj – una señal de reloj
  - Reset (restaurar) – una señal de inicialización

# 1.5 CLASIFICACION DE ARQUITECTURAS

Instruction Set Architecture (ISA)

ISA, es la parte de la arquitectura de computadores en relación con la programación, incluídos los tipos de datos nativos, instrucciones, registros, modos de direccionamiento, la arquitectura de memoria, manejo de interrupciones y excepciones, y E/S externa.

Una ISA incluye una especificación del conjunto de códigos de operación (lenguaje máquina), los commandos nativos implementados por un diseño particular de la CPU.

Las Instrucciones incluyen:

- \* Instrucciones aritméticas tales como sumar y restar
- \* Instructions Lógicas tales como and, or, y not
- \* Instrucciones de datos tales como move, input, output, load, y store
- \* Instructions de control de flujo tales como goto, if ... goto, call, y

# USO BASICO DEL WATCHDOG TIMER EN ARDUINO

```
/*  
 *  
 *  Blink  
 *  
 *  Turns on an LED on for one second, then off for one second, repeatedly.  
 *  
 *  This example code is in the public domain.  
 */  
  
// Pin 13 has an LED connected on most Arduino boards,  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)  
    delay(1000);              // wait for a second  
    digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW  
    delay(1000);              // wait for a second  
}
```

Comentarios o descripción

Declaración de Variables

Sección de configuración

Sección del bucle

# USO BASICO DEL WATCHDOG TIMER

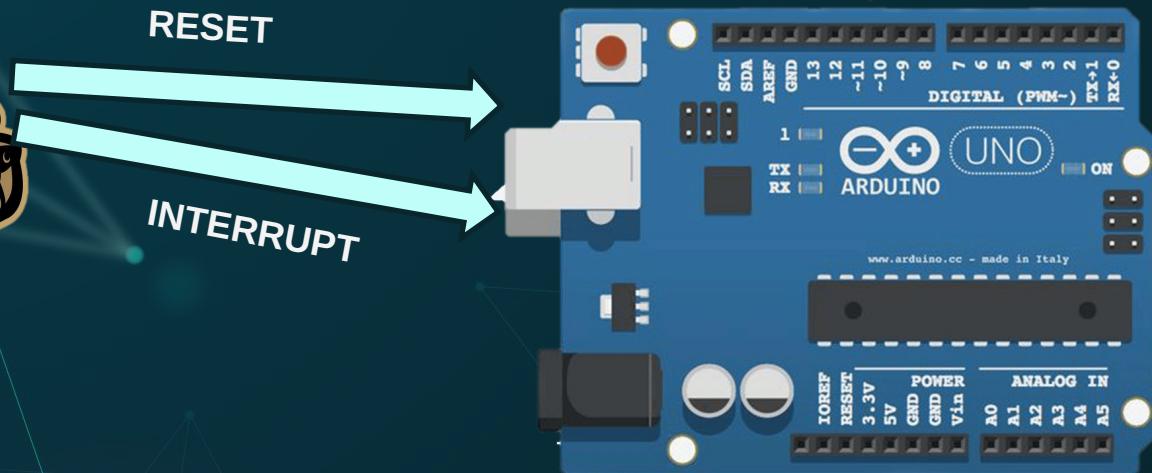
## EN ARDUINO

Watchdog Timer: Se utiliza para protegerse contra errores en el programa y/o el hardware. El temporizador de vigilancia se utiliza para reiniciar el controlador en caso de bloqueo del software.



RESET

INTERRUPT



# USO BASICO DEL WATCHDOG TIMER

Watchdog timer library **EN ARDUINO**

```
#include <avr/wdt.h>
```

# USO BASICO DEL WATCHDOG TIMER EN ARDUINO

`wdt_disable():` // comando para deshabilitar el watchdog timer  
`wdt_enable(WDTO_4s)` // configuración de tiempo  
`wdt_reset();` // comando reseteador del reloj del watch dog timer

# USO BASICO DEL WATCHDOG TIMER

## EN ARDUINO

Watchdog timer library

```
wdt_enable(WDTO_4s) // configuración de tiempo
```

Argumento de wtd_enable ()	Tiempo antes que se dispare el watchdog
WDTO_15MS	15mS
WDTO_30MS	30mS
WDTO_60MS	60mS
WDTO_120MS	120mS
WDTO_250MS	250mS
WDTO_500MS	500mS
WDTO_1S	1S
WDTO_2S	2S
WDTO_4S	4S
WDTO_8S	8S

# SISTEMAS EMBEBIDOS I

## UNIDAD 2 – MEMORIAS

ELMER ALAN CORNEJO QUITO

## 2. MEMORIAS

2.1 Tipos de memoria de un microcontrolador

2.2 Acceso de memoria de un microcontrolador

2.3 Segmentación de memoria

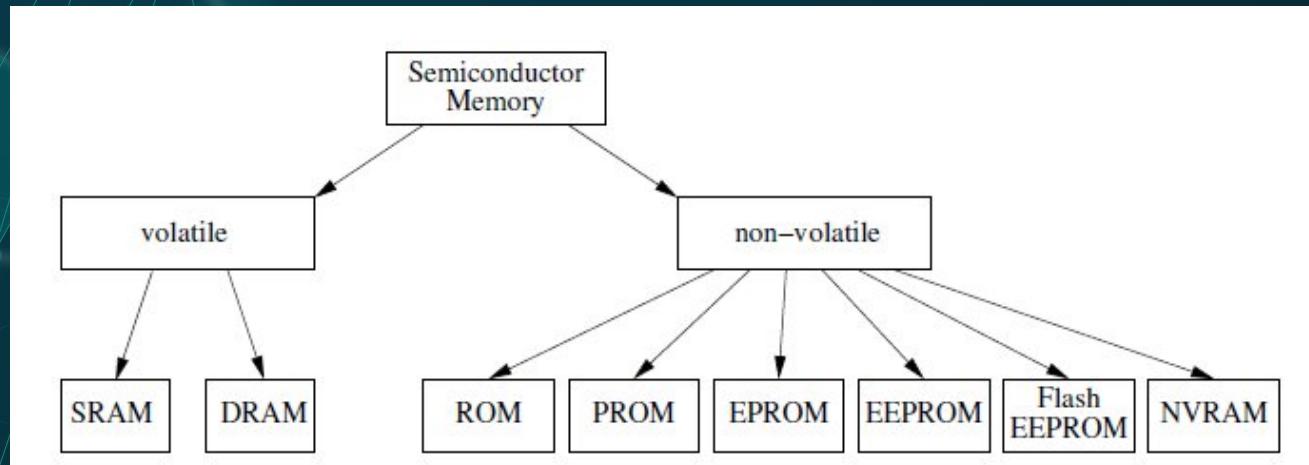
2.4 Manipulación de Memoria

## 2. MEMORIAS

- Register file (Archivo de registro): Usualmente una memoria pequeña encapsulada en el CPU, utilizada para almacenamiento temporal de valores del CPU. La memoria a corto plazo de la CPU.
- 1. Data Memory (Memoria de datos): Para almacenaje a largo plazo. Las CPUs normalmente utilizan este tipo de memorias externas, que son mas grandes que las anteriores. Los microcontroladores ya vienen con estas memorias encapsuladas en un solo chip.
- 2. Instruction Memory (Memoria de instrucciones): Memoria externa a los CPUs, integrada en los MCUs. Memoria relativamente grande se utiliza para almacenaje de instrucciones.

## 2. MEMORIAS

1. Las memorias en los microcontroladores, pueden clasificarse a su vez en:



## 2. MEMORIAS

### Memoria volátil

- Este tipo de memoria es usualmente mas rápida que la memoria no volátil.
- En cuanto se quita la energía al equipo, la información almacenada es perdida.
- Por razones históricas las memorias volátiles son generalmente llamadas

No significa que la memoria sea capaz de acceder al azar a alguna parte de la memoria, sino que desde la perspectiva de la memoria, cualquier dirección, puede ser accedida.



Antes se tenían distintos tipos de memorias volátiles: las que permitían acceso a cualquier dirección y las que eran de acceso secuencial (llamadas "shift register memory").

## 2. MEMORIAS

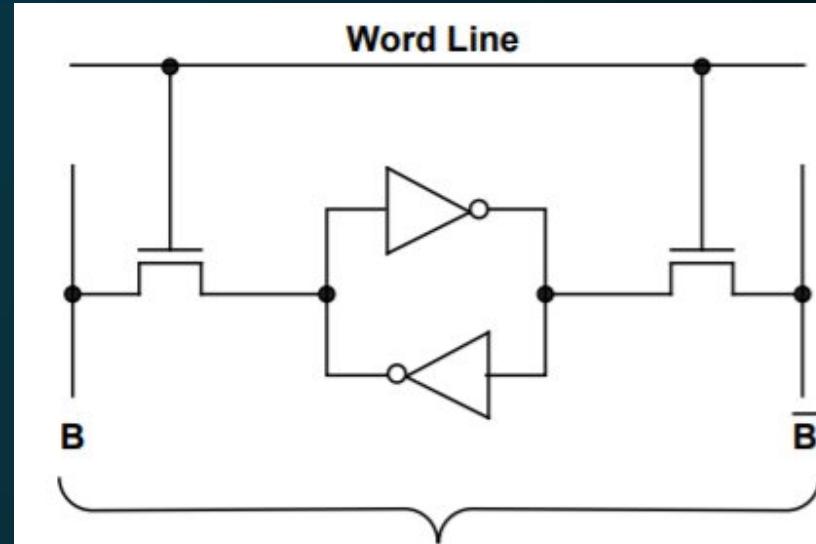
### Memoria volátil

**-SRAM, Static RAM:** Basada en semiconductores, almacena los datos de una manera estática. No necesita ser actualizada de forma dinámica como ocurre en las DRAM.

-Pierde la información almacenada una vez que se quite la energía al componente.

-Provee velocidades que no se pueden lograr con las DRAM.

-Sirve en muchos casos como memoria caché entre la CPU y la DRAM.



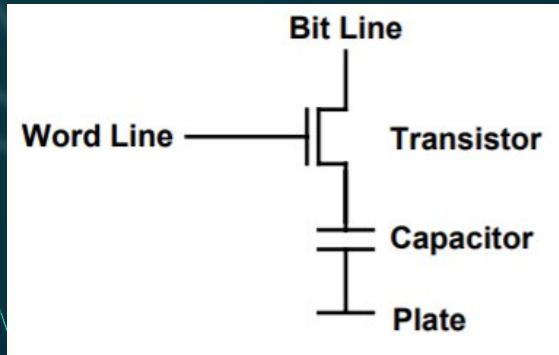
Source: ICE, "Memory 1997"

20019

Figure 8-3. SRAM Cell

## 2. MEMORIAS

DRAM, Dinamyc RAM: Cada celda de almacenaje de memoria está conformada por un mosfet y un capacitor. La carga almacenada en el capacitor se filtra, es por esto que se requiere de un ciclo de refresco varias veces cada segundo.



Source: ICE, "Memory 1997"

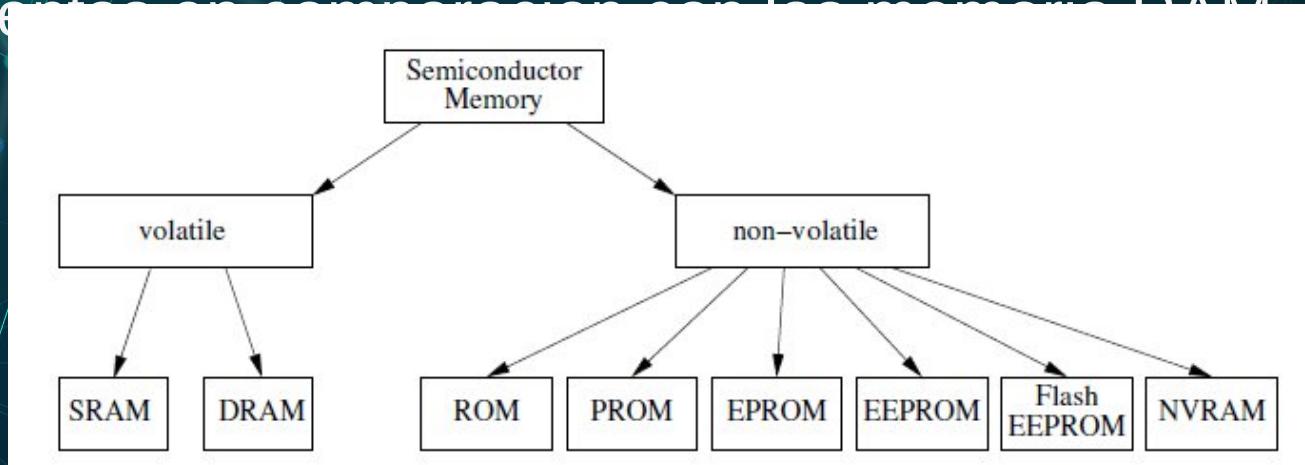
19941

Figure 7-1. DRAM Cell

## 2. MEMORIAS

Memoria no volátil

- A diferencia de las SRAM y las DRAM, las memorias no volátiles retienen su contenido incluso cuando se corta la energía.
- Son le



## 2. MEMORIAS

**Memoria ROM Read Only Memory:** El acceso a estas memorias es de solo lectura, puede ser leída pero sobre la cual no se puede destruir o reemplazar algo.

- Utilizado principalmente para almacenar el firmware de un equipo.
- En un sentido más estricto, ROM hace referencia a “mask ROM”, que es el tipo más antiguo de ROM de estado sólido (solid state ROM). Este era fabricado con el almacenamiento, de manera permanente, de datos deseados y no podía ser modificado.

## 2. MEMORIAS

Memoria PROM: Programmable Read Only Memory, son memorias ROM que pueden ser programadas solo una vez.

- Básicamente son matrices de celdas de memoria, cada una con un fusible, al pasar un pulso de corriente alto por este, puede ser destruido y de esa forma programar un 0 lógico en la celda seleccionada.
- Existen MCUs que son OTP (One Time Programmable), que contienen PROMs como memoria de instrucción.

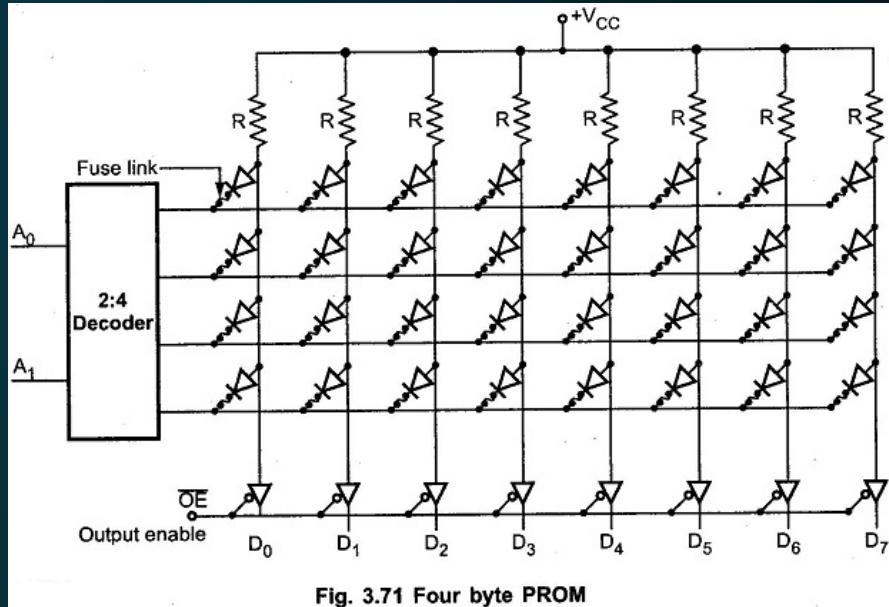


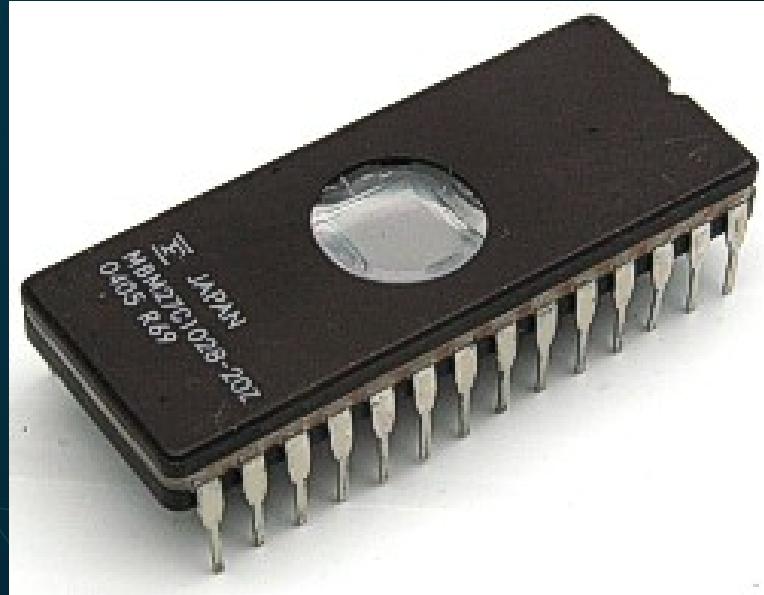
Fig. 3.71 Four byte PROM

## 2. MEMORIAS

### Memoria EPROM: Erasable

Programmable Read Only Memory, son memorias PROM que se pueden borrar.

- Los datos son almacenados por medio de FETs. Una vez la celda está programada los datos se guardan en un periodo de alrededor 10 años.
- Para poder borrar la memoria, se remueve la capa protectora con la que cuentan y se expone el chip a luz ultravioleta.



## 2. MEMORIAS

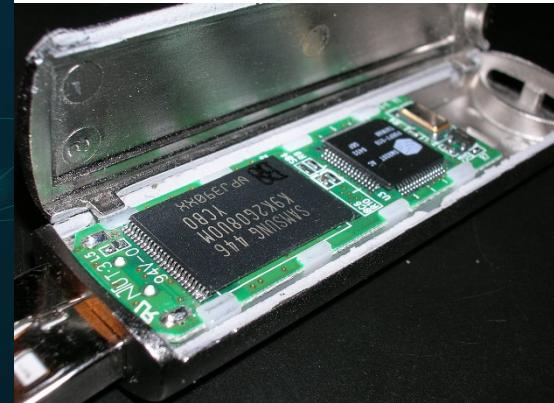
### Memoria EEPROM:

- Electrical Erasable Programmable Read Only Memory, también memorias PROM borrables.
- Memoria borrible aplicando un alto voltaje.
- Estas memorias son capaces de ser borradas/escritas un número limitado de veces, usualmente en el orden de los 100 000 veces. Y el tiempo de retención de la memoria tampoco es permanente, igual que con las EPROM.
- Debido a su limitado número de escrituras/borradas estas memorias son utilizadas para almacenaje de información a largo plazo.

## 2. MEMORIAS

### Memoria Flash (Flash EEPROM)

- Variante de la EEPROM, donde el borrado o la reprogramación no es posible de realizar dirección por dirección, sino múltiples direcciones al mismo tiempo.
- Este tipo de memorias es utilizado comúnmente para programas y no así para datos.
- Empleada en los dispositivos denominados memoria USB.



## 2. MEMORIAS

### Memoria NVRAM

- Combina las ventajas de las memorias volátil y no volátil.
- Se puede lograr juntar características de las memorias volátiles y no volátiles de varias formas. Una de ellas es agregando una pequeña batería a un dispositivo SRAM, así cuando la energía se corte la información estará retenida en la SRAM.
- Otra es juntando una SRAM y una EEPROM.



## 2. MEMORIAS

### SDRAM: (Synchronous RAM)

- Tipo de memoria DRAM que funciona de manera sincronizada.
- Sincronizada con la velocidad del reloj con la que el CPU esta optimizado.
- Esto permite una transferencia de datos a mayor velocidad.
- Los DRAM usan interfaces asíncronas, donde las señales de control de entrada llegan al CPU pero de forma retrasada al no estar sincronizados sus tiempos de acción.
- En cambio las señales de control de entrada de las SDRAM llegan directamente al CPU.

## 2. MEMORIAS

### DDR SDRAM: (Double Data Rate SDRAM)

- Lo que la diferencia de la SDRAM no es la velocidad de trasferencia, sino que el monto de memoria que maneja en la transferencia de datos, que es mayor.
- Realiza la transferencia de datos en dos tiempos, a la subida y bajada del ciclo del reloj de la señal.

# ARDUINO (ATMEGA 328)

## Memoria Flash

Donde se guarda el programa, el sketch de Arduino que se vaya a crear.

No volátil.

Tiene un tamaño de 32 KB, de los cuales 0.5 KB son utilizados por el bootloader.

## SRAM

Volátil.

Lugar donde los programas almacenan y manejan sus variables. Memoria de datos.

Tiene un tamaño de 2 KB (2048 Bytes).

## EEPROM

No volátil.

Para almacenar información a largo plazo.

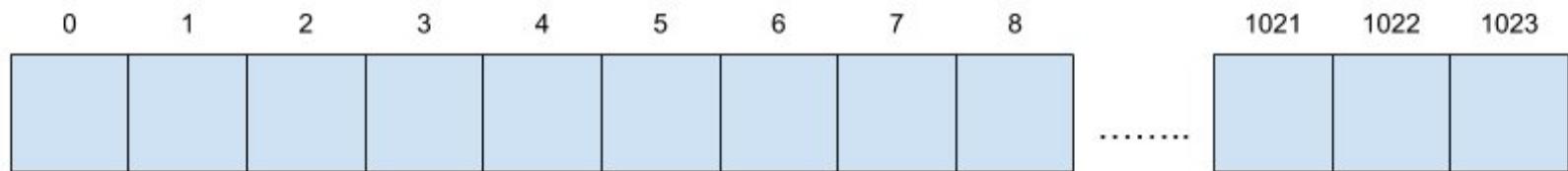
Tiene un tamaño de 1 KB (1024 Bytes).

# MEMORIAS EEPROM EN ARDUINO

MICROCONTROLADOR	EEPROM
ATmega328 (Arduino UNO, Nano, Mini)	1024 bytes
ATmega168 (Arduino Nano)	512 bytes
ATmega2560 (Arduino Mega)	4096 bytes

# EEPROM EN ARDUINO

En el caso de Arduino UNO, el microcontrolador ATmega328 tiene 1024 bytes, es decir, 1024 posiciones de memoria.



# EEPROM EN ARDUINO

```
#include <EEPROM.h>
```

# EEPROM EN ARDUINO

**EEPROM Clear:** Clear the bytes in the EEPROM.

**EEPROM Read:** Read the EEPROM and send its values to the computer.

**EEPROM Write:** Stores values from an analog input to the EEPROM.

**EEPROM Crc:** Calculates the CRC of EEPROM contents as if it was an array.

**EEPROM Get:** Get values from EEPROM and prints as float on serial.

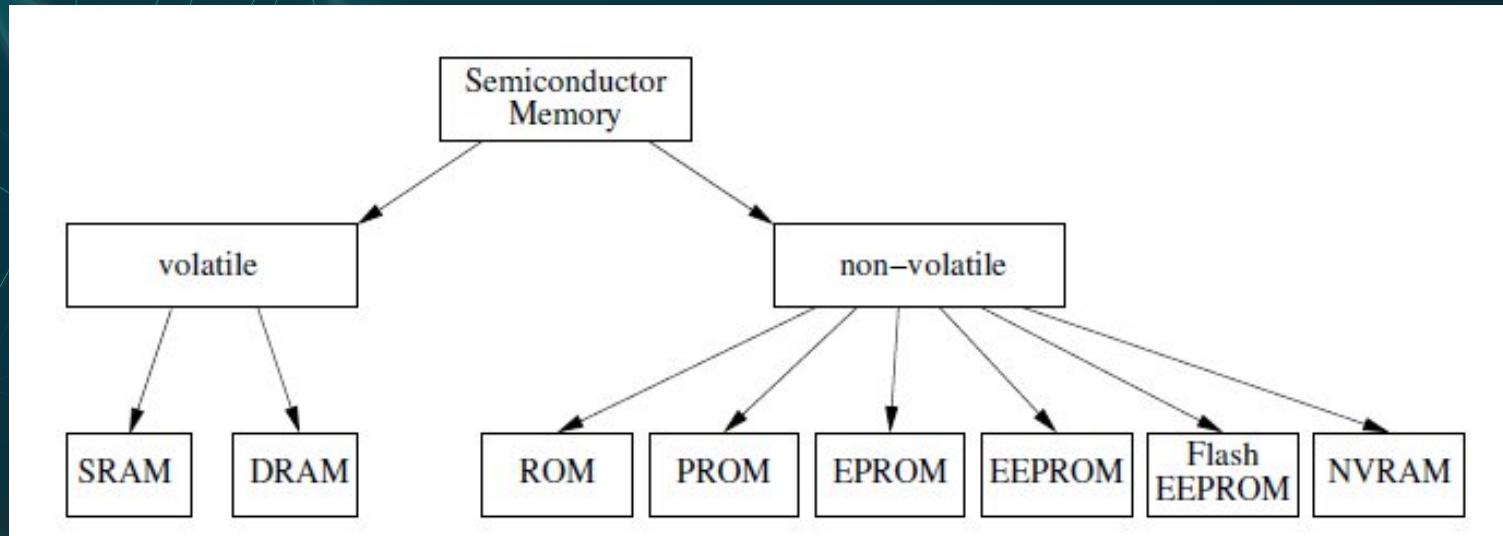
**EEPROM Iteration:** Understand how to go through the EEPROM memory locations.

**EEPROM Put:** Put values in EEPROM using variable semantics.

**EEPROM Update:** Stores values read from A0 into EEPROM, writing the value only if different, to increase EEPROM life.

## 2.2 Acceso de memoria de un microcontrolador

Muchos microcontroladores vienen con memoria de programas y memoria de datos.



# USO BASICO DEL WATCHDOG TIMER EN ARDUINO

Ejemplos para Arduino Mega or Mega 2560

EEPROM

SoftwareSerial

SPI

Wire

Ejemplos de Librerías Personalizadas

Adafruit FONA Library

Adafruit MQTT Library

Adafruit SleepyDog Library

eeprom\_clear

eeprom\_crc

eeprom\_get

eeprom\_iteration

eeprom\_put

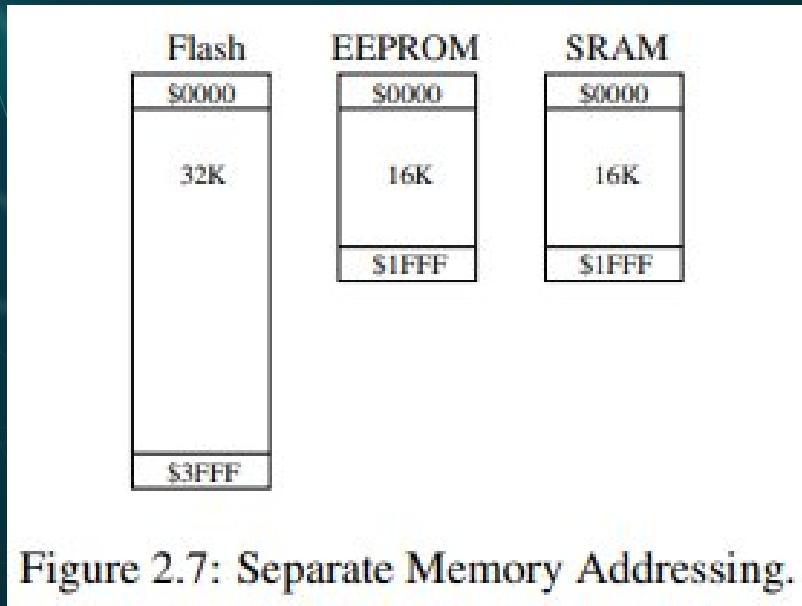
eeprom\_read

eeprom\_update

eeprom\_write

## 2.2 Acceso de memoria de un microcontrolador

Cada memoria se direcciona por separado



## 2.2 Acceso de memoria de un microcontrolador

Los rangos de direcciones de los tres tipos de memoria diferentes pueden ser los mismos

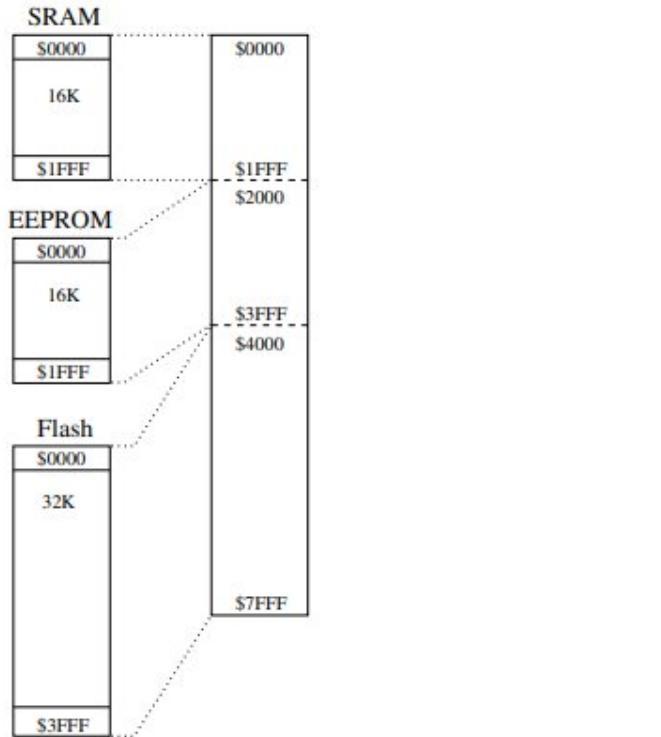


Figure 2.8: Different memory types mapped into one address range.

## 2.2 Acceso de memoria de un microcontrolador

Los rangos de direcciones de los tres tipos de memoria diferentes pueden ser los mismos

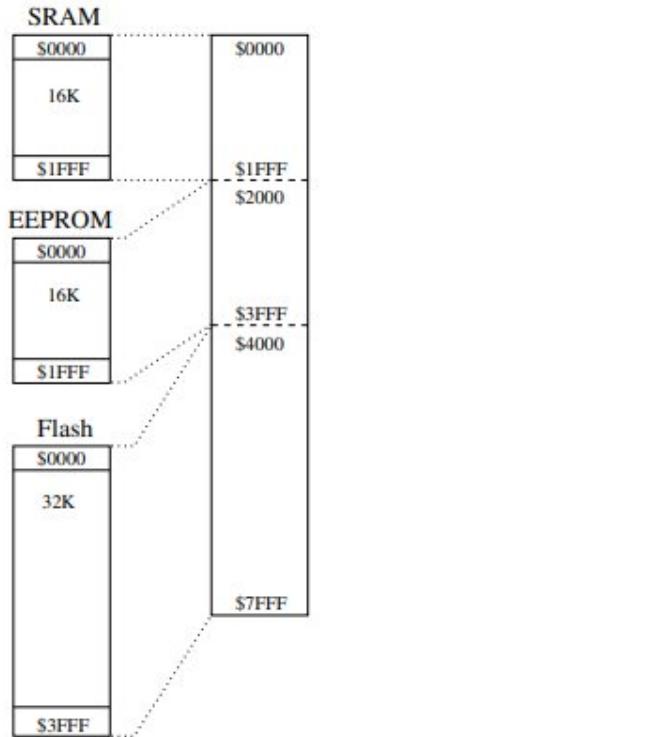


Figure 2.8: Different memory types mapped into one address range.

# Unidad Temática 3: Periféricos

3.1 Digital IO

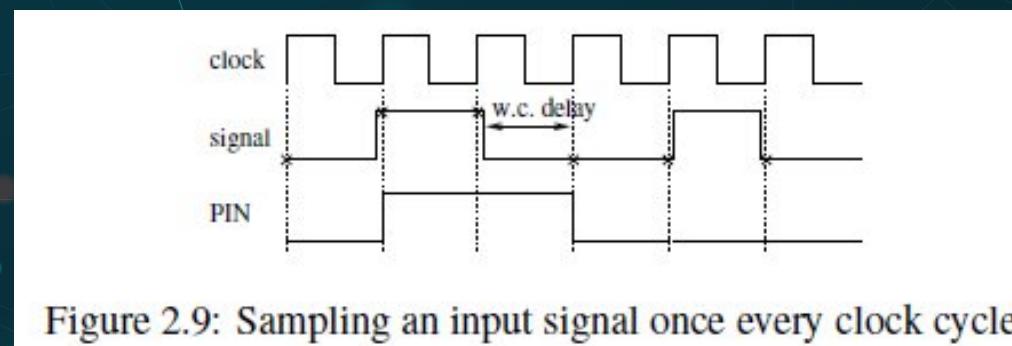
3.2 ADC y DCA

3.3 PWM y TIMERS

3.4 Clock

# 3.1 Digital IO

- Es el medio por el cual se controla hardware y dispositivos externos al MCU.
- Estos pines pueden ser solo de entrada o solo de salida, o como en la mayoría de los casos, bidireccionales.
- Al ser estos pines digitales, solo se puede enviar o recibir dos valores diferentes, encendido o apagado, ON/OFF.
- Para las entradas digitales, se debe tener en cuenta la velocidad del reloj con la que se trabaja para captar la señal que se esta recibiendo.



# 3.1 Digital IO

- Muchos MCUs tienen integrado un circuito *pull resistor* a sus entradas para reducir el ruido de la interferencia electromagnética, que se genera en el ambiente.
- Estos pueden ser, *pull up* o *pull down* para deshacerse del ruido.
- En cuanto a las salidas digitales, estas son usadas para poner el pin deseado a dos posibles valores, HIGH/LOW, 1-0, ON/OFF.

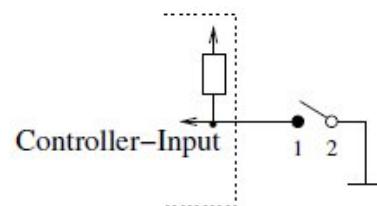
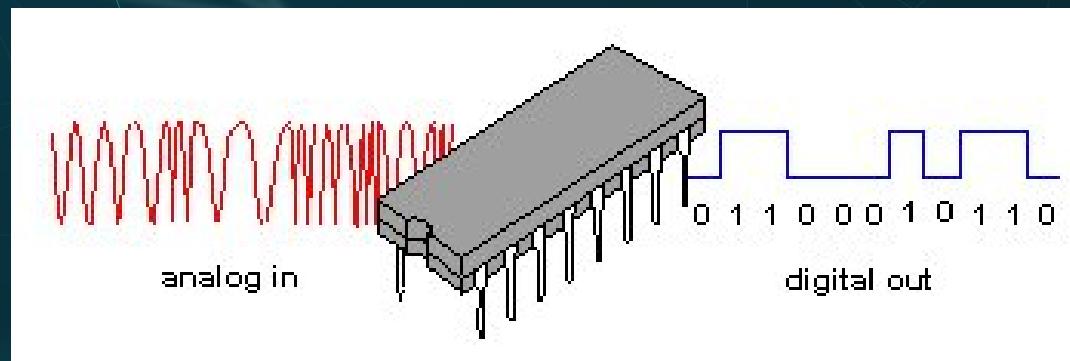


Figure 2.12: Attaching a switch to an input pin with activated pull-up resistor.

# 3.1 Analogic IO

- Si se quiere medir el voltaje de un foto transistor, este no solo presentará dos valores, por lo que se requiere una forma de captar este tipo de señales, analógicas.
- Se hace uso del conversor Analógico/Digital (A/D) dispositivo electrónico capaz de convertir una señal analógica en un valor binario.
- La resolución determina la precisión con la que se reproduce la señal original.
  - $\text{Resolución} = +V_{ref}/2^n$  (donde  $n$  son bits)

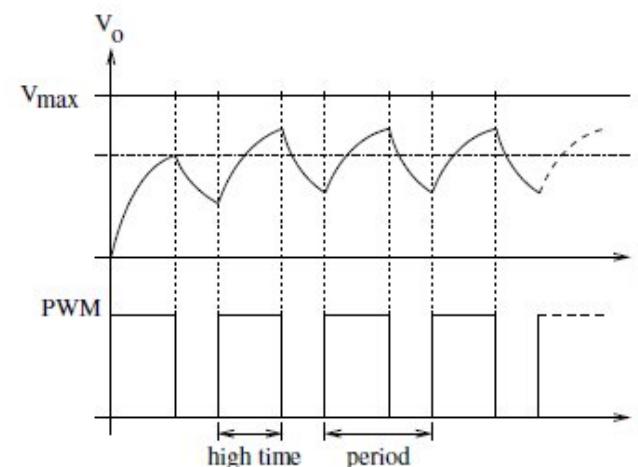
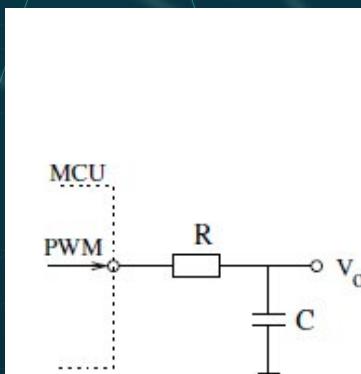


## 3.1 Analogic IO

- Si se quiere medir el voltaje de un foto transistor, este no solo presentará dos valores, por lo que se requiere una forma de captar este tipo de señales, analógicas.
- Se hace uso del conversor Analógico/Digital (A/D) dispositivo electrónico capaz de convertir una señal analógica en un valor binario.
- La resolución determina la precisión con la que se reproduce la señal original.
  - *Resolución = +Vref/2^n* (donde n son bits)
  - Por ejemplo, un conversor A/D de 8 bits su resolución será: Vref/256.
  - Mapeará los valores de voltaje de entrada, entre 0 y Vref voltios, a valores enteros comprendidos entre 0 y 255 ( $2^n-1$ ).

### 3.1 Analogic IO

- Para la salida analógica, muchos de los microcontroladores implementan un conversor D/A, haciendo uso de las señales PWM (Pulse-Width Modulation). Al mismo tiempo se puede mejorar la señal analógica de salida acoplando un circuito de filtro pasa bajo RC .



# ATMEGA 328

## E/S pines

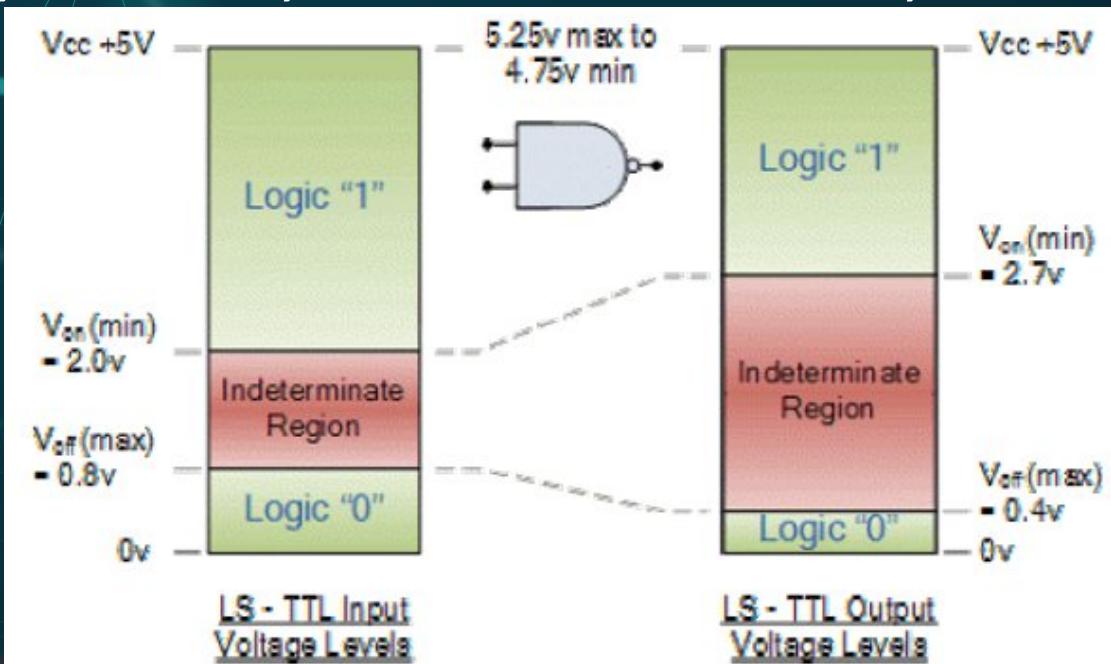
- 23 pines E/S programables
- En la placa Arduino UNO, 14 pines están dedicados a las E/S digitales. 6 de ellas son de tipo PWM.
- 6 pines dedicados a entradas analógicas.
- Los demás 3 pines están destinados al botón de RESET y los dos osciladores TOSC1 y TOSC2.
- El voltaje de operación es de 1,8 a 5,5 V.

(PCINT14/RESET) PC6	1	28	□ PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	□ PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	□ PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	□ PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	□ PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	□ PC0 (ADC0/PCINT8)
VCC	7	22	□ GND
GND	8	21	□ AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	□ AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	□ PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	□ PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	□ PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	□ PB2 (SS/OC1B/PCINT2)
(PCINT0/CLK0/ICP1) PB0	14	15	□ PB1 (OC1A/PCINT1)

# ATMEGA 328

## E/S pines

Si se emplea lógica TTL donde un 0 se representa por un voltaje entre 0 y 0,8 Volts y un valor HIGH entre 2 y 5 Volts.

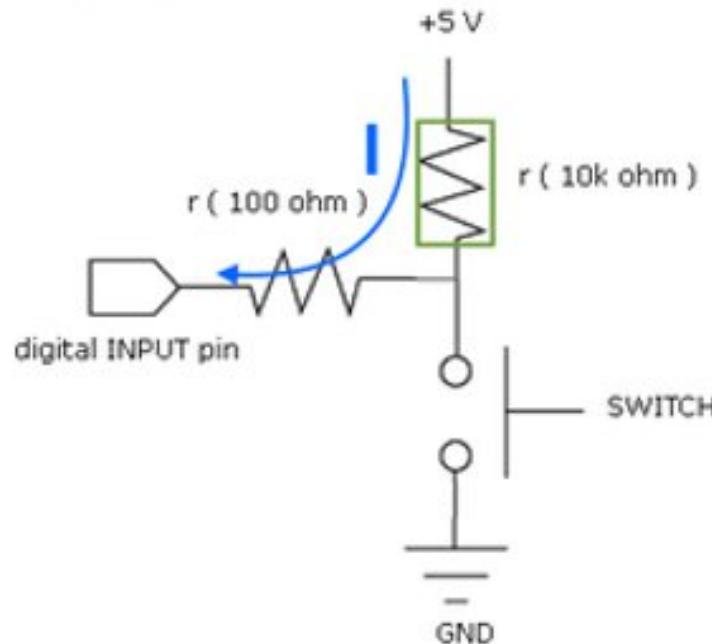


# ATMEGA 328

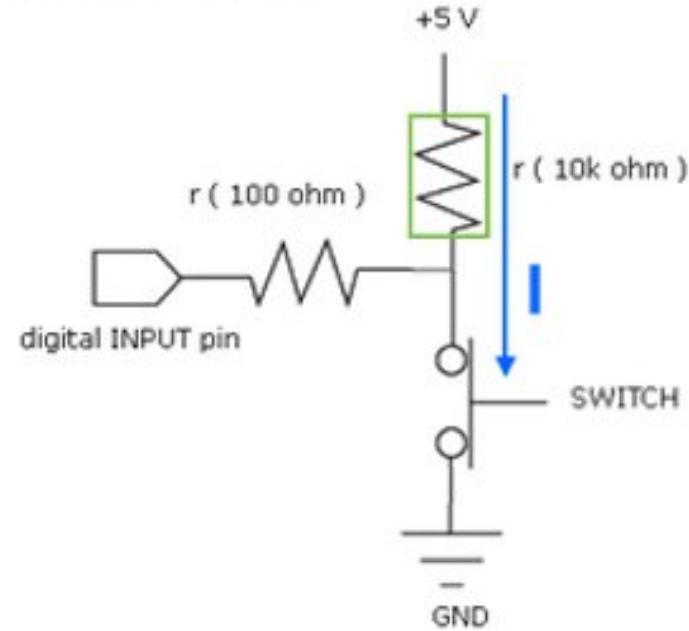
E/S pines

- Pull-up resistor:

Switch with "pull-up" resistor



Switch with "pull-up" resistor

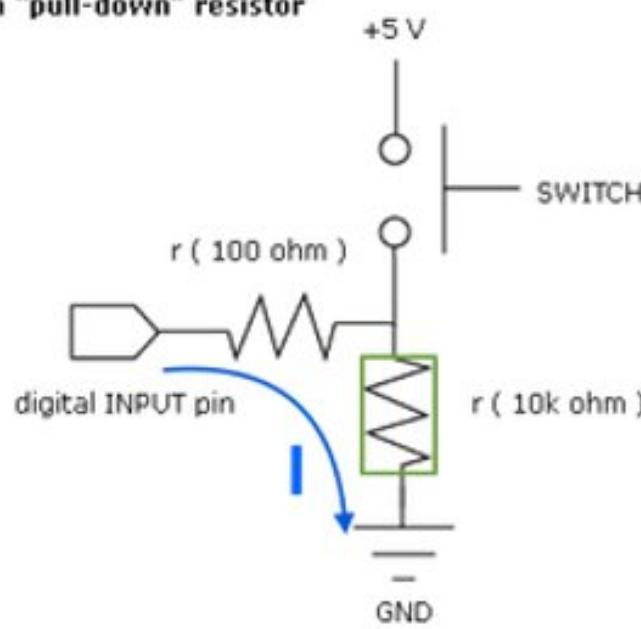


# ATMEGA 328

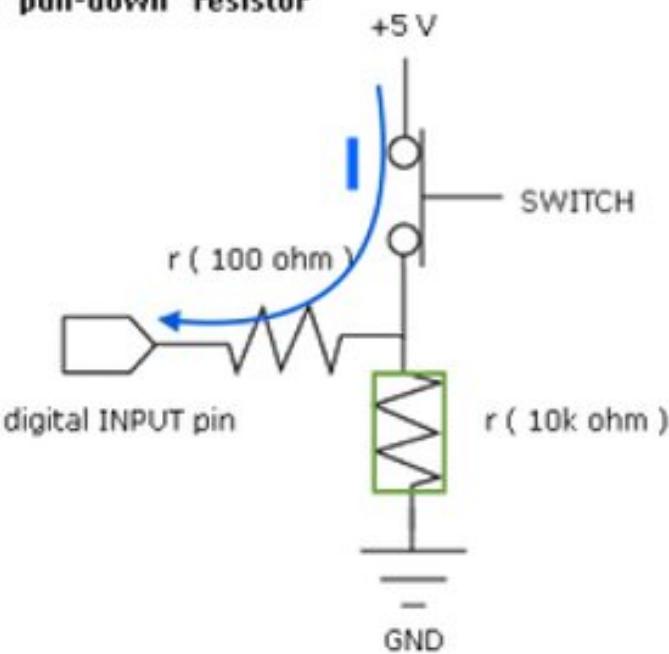
E/S pines

Pull-down resistor:

Switch with "pull-down" resistor



Switch with "pull-down" resistor



# ATMEGA 328

## E/S pines

Existe también la opción de habilitar resistores pull-up internos del microcontrolador ATmega328p. Esto se puede hacer mediante comandos en el IDE de Arduino.

```
int pinEntrada = 2;

void setup() {
    // put your setup code here, to run once:
    pinMode(pinEntrada, INPUT_PULLUP);
    Serial.begin(9600);

}

void loop() {
    // put your main code here, to run repeatedly:
    if (digitalRead(pinEntrada) == HIGH)
    {
        Serial.println("pulsado");
    }
    else
    {
        Serial.println("nada");
    }
}
```

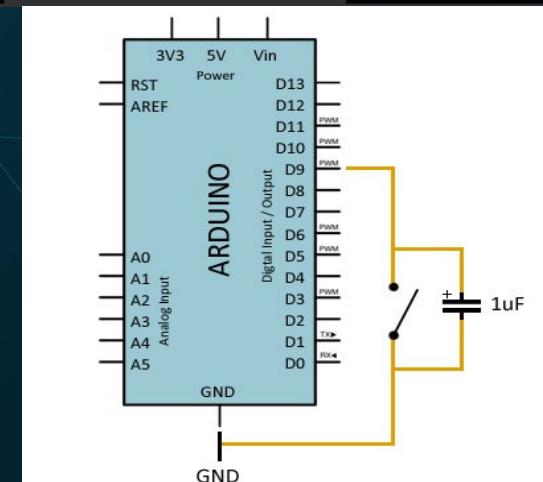
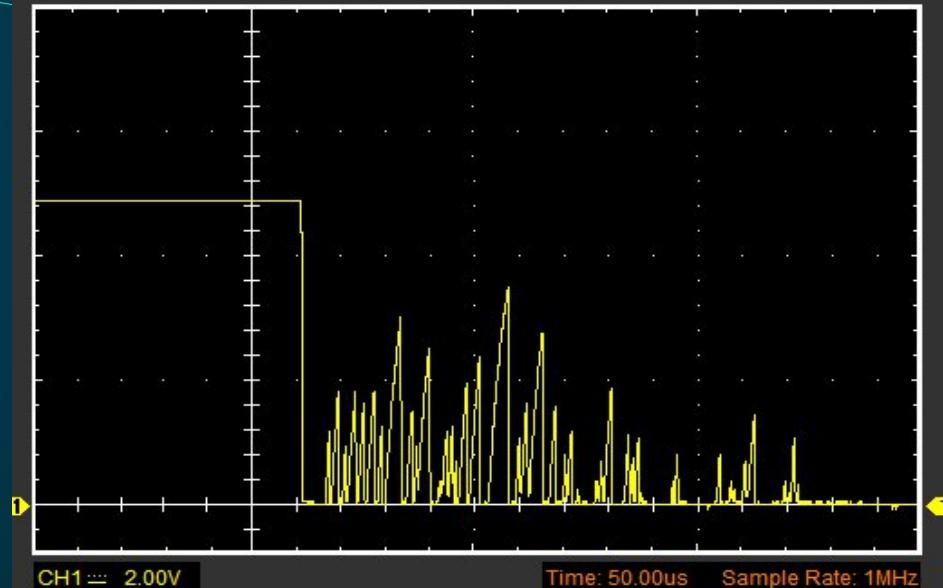
# ATMEGA 328

E/S pines

## EL REBOTE

Muchos dispositivos físicos habitualmente generan ruido en los flancos de señal. Como ejemplo, la variación de tensión que ocurre cuando el cambio de estado se genera por un pulsador.

Eliminar rebote por hardware:



# INTERRUPCIONES

- Se detiene la ejecución normal del programa para realizar una acción en específico. Una vez terminada dicha acción, se vuelve al flujo normal del programa.
- En los MCUs se tienen dos bits para manejar las interrupciones.
- IE (Interrupt Enable), para indicar que el controlador debe llamar a ISR (Interrupt Service Routine), como reacción al evento.
- IF (Interrupt Flag), es activado cuando el evento de interrupción ocurre y es limpiado automáticamente cuando se entra al ISR.
- El llamado de una interrupción sigue los siguientes pasos:
  - Activar el IF
  - Finaliza la presente instrucción
  - Identifica el ISR
  - Llama al ISR

# ATMEGA 328

- Se tiene interrupciones con timers, (por software). O las interrupciones por hardware, que responden a los siguientes eventos.
  - RISING, ocurre en el flanco de subida de LOW a HIGH.
  - FALLING, ocurre en el flanco de bajada de HIGH a LOW.
  - CHANGING, ocurre cuando el pin cambia de estado (rising + falling).
  - LOW, se ejecuta continuamente mientras está en estado LOW.

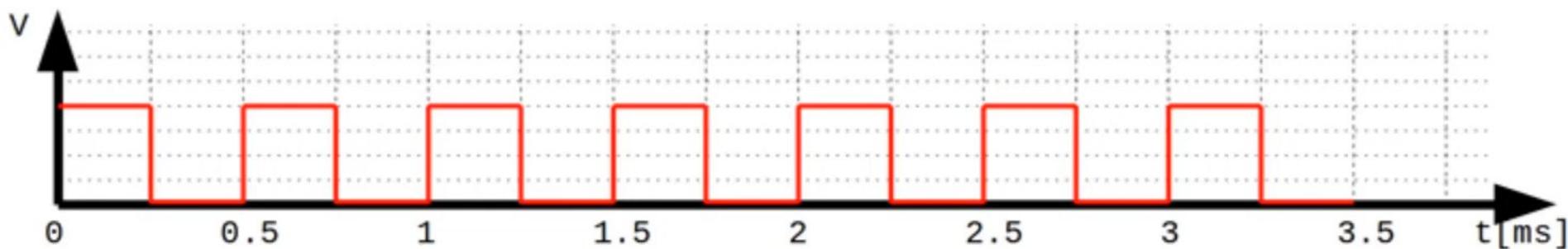
Modelo	INT0	INT1	INT2	INT3	INT4	INT5
UNO	2	3				
Nano	2	3				
Mini Pro	2	3				
Mega	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due	En todos los pines					

# INTERRUPCIONES EN ARDUINO

- La función que maneja una interrupción es la ISR (Interrupt Service Routines).
- Normalmente se limita a incrementar un contador o modificar una variable.
- Si una variable es utilizada dentro de la función ISR y en bucle principal, debe ser declarada como “*volatile*”.
- Específicamente se ordena al compilador a cargar esta variable desde la memoria RAM y no desde el registro de almacenamiento, la cual es una memoria temporal donde se manipulan las variables.

# Clock (Reloj)

Círcuito electrónico que genera una onda cuadrada con un determinado periodo en otras palabras genera pulsos constantes en determinados tiempos



# INTERRUPCIONE EN ARDUINO

- Programa que cuenta los pulsos enviados, por medio de interrupciones.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** InterrupcionesEjemploBasico | Arduino 1.6.8
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Open, Save, Undo, Redo, and Upload/Download.
- Code Editor:** Displays the following C++ code for an Arduino sketch:

```
const int emuPin = 10;
const int intPin = 2;
volatile int cont = 0;

void setup() {
    pinMode(emuPin, OUTPUT);
    pinMode(intPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(intPin), counter, RISING);
    Serial.begin(9600);
}

void loop() {
    //esta parte es para emular la salida
    digitalWrite(emuPin, HIGH);
    delay(1000);
    digitalWrite(emuPin, LOW);
    delay(1000);
    Serial.println("valor del contador:");
    Serial.println(cont);
}

void counter() {
    cont++;
}
```

# TIMER (TEMPORIZADOR)

- La mayoría de los MCUs cuentan con uno o varios timers con 8 y/o 16 bits de resolución.
- Se lo utiliza principalmente para medir tiempos con la precisión que nos permita el MCU.
- Cumplen con distintas tareas:
  - Contadores.
  - Medida de periodos.
  - Generación de retrasos.
  - Generación de formas de onda.

# ATMEGA 328p

- Timers
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode.
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode.
  - En la placa Arduino, estos se llaman Timer0, Timer1 y Timer2.
  - Las funciones de Arduino `delay()`, `millis()` and `micros()` and `delayMicroseconds()` utilizan timers por detrás.
  - `analogWrite()`, `tone()` y `noTone()` utilizan también los timers. Hasta la librería de servo utiliza timers.
  - Es como un reloj que se utiliza para medir el tiempo de determinados eventos.
  - Todos los timers dependen del reloj del sistema, que en este caso es de 16 MHz.

# ATMEGA 328p

- Timer0
  - Timer de 8 bits, utilizado para las funciones: delay(), millis() and micros().
- Timer1
  - Timer de 16 bits, utilizado para las funciones de la librería servo.
- Timer2
  - Timer de 8 bits, utilizado para las funciones de tone().
    - Genera una onda cuadrada de la frecuencia especificada (y un ciclo de trabajo del 50%) en un pin.

# Millis y Micros

CODIGO-01 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

CODIGO-01

```
1 void setup() {  
2     Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6     Serial.println(millis());  
7     delay(500);  
8 }
```

CODIGO-02 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

CODIGO-02

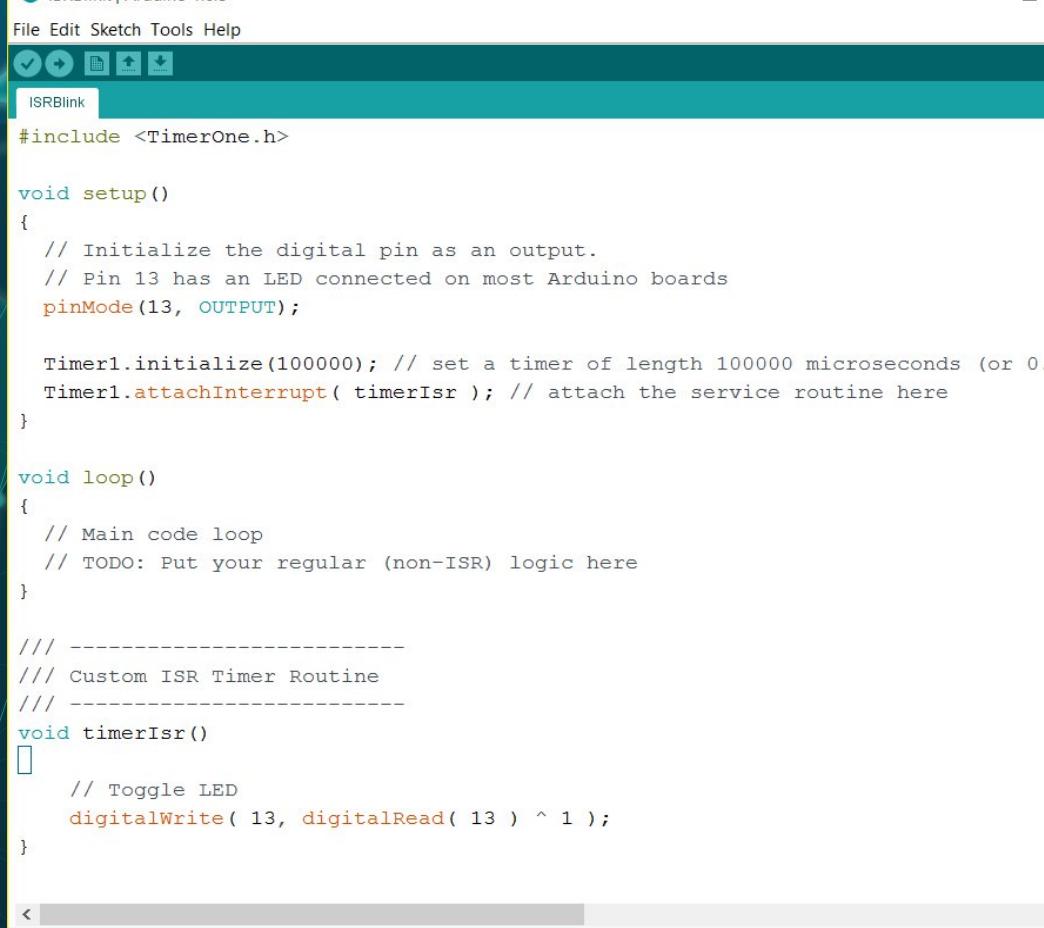
Subir

```
1 void setup() {  
2     Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6     Serial.println(micros());  
7     delay(500);  
8 }
```

# Clock (Reloj)

Tipos de datos	Valor mínimo	Valor máximo
byte	0	255
integer (int)	-32768	32767
unsigned int	0	65535
long	-2147483648	2147483647
unsigned long	0	4294967295

# Timer One



The image shows a screenshot of the Arduino IDE. The title bar reads "File Edit Sketch Tools Help". The menu bar includes icons for File, Edit, Sketch, Tools, and Help. The toolbar below the menu bar has icons for Open, Save, Print, and Upload. The sketch name "ISRBlink" is displayed in the title bar. The code editor contains the following C++ code:

```
File Edit Sketch Tools Help
ISRBlink
#include <TimerOne.h>

void setup()
{
    // Initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards
    pinMode(13, OUTPUT);

    Timer1.initialize(100000); // set a timer of length 100000 microseconds (or 0.1 second)
    Timer1.attachInterrupt( timerIsr ); // attach the service routine here
}

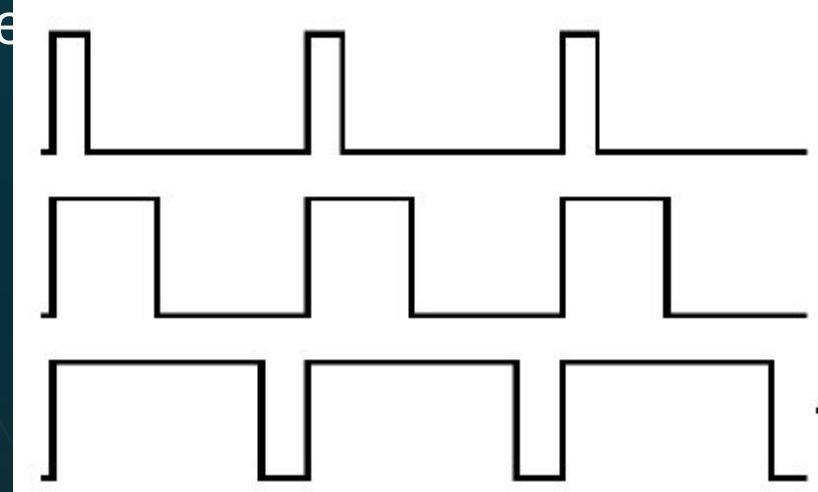
void loop()
{
    // Main code loop
    // TODO: Put your regular (non-ISR) logic here
}

/// -----
/// Custom ISR Timer Routine
/// -----
void timerIsr()
{
    // Toggle LED
    digitalWrite( 13, digitalRead( 13 ) ^ 1 );
}
```

# PWM- Pulse Width Modulation (Modulación por ancho de banda )

Una señal PWM es una forma de onda digital binaria de una determinada frecuencia y ciclo de trabajo variable.

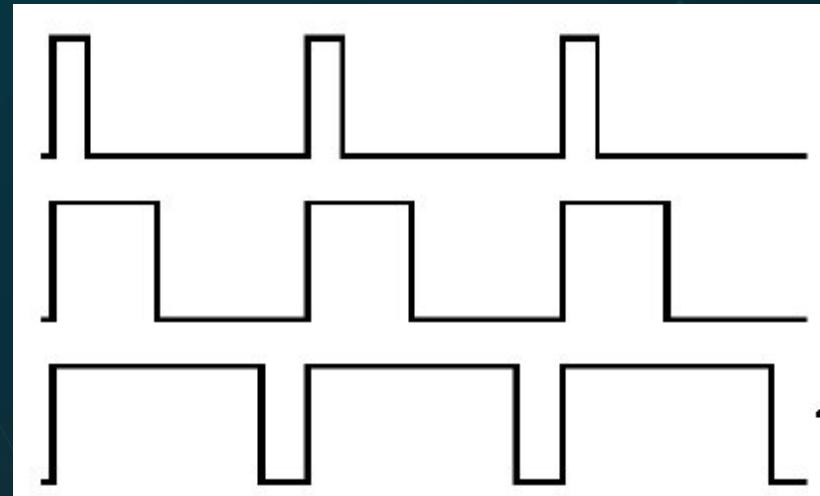
PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. En esta técnica se modifica el ciclo de trabajo de una señal pe



# PWM- Pulse Width Modulation (Modulación por ancho de banda )

USOS:

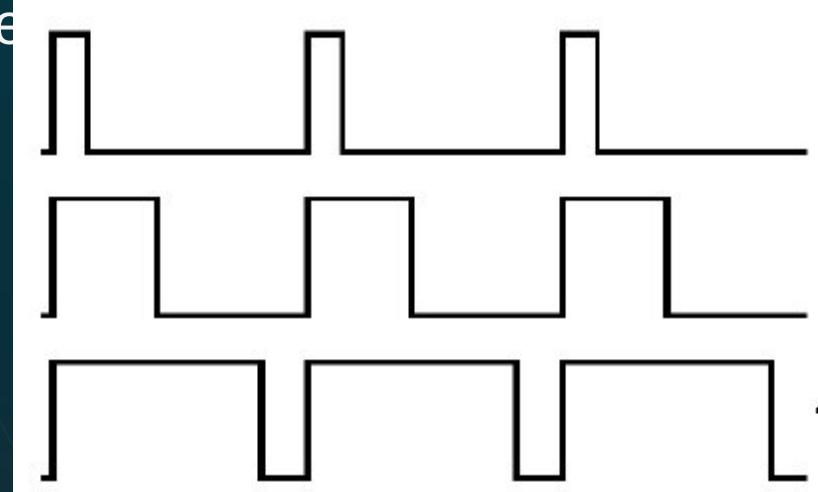
- controlar la cantidad de energía que se envía a una carga
- regular la velocidad de giro de los motores
- regulación de intensidad luminosa



# PWM- Pulse Width Modulation (Modulación por ancho de banda )

Una señal PWM es una forma de onda digital binaria de una determinada frecuencia y ciclo de trabajo variable.

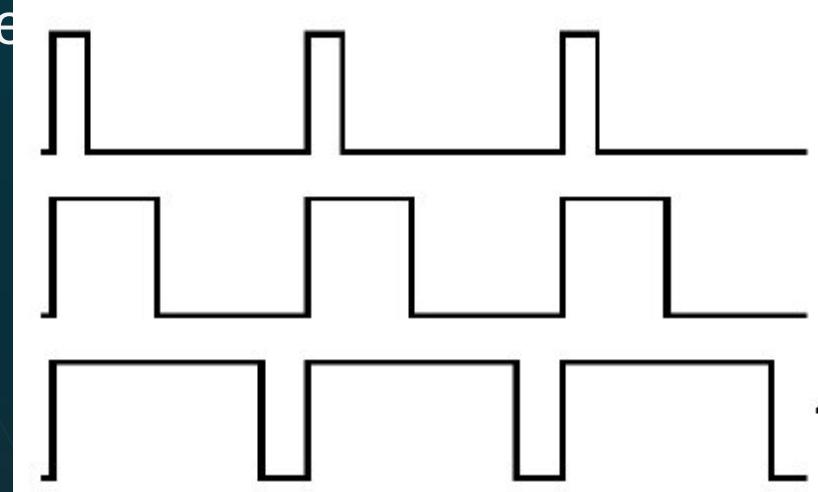
PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. En esta técnica se modifica el ciclo de trabajo de una señal pe



# PWM- Pulse Width Modulation (Modulación por ancho de banda )

Una señal PWM es una forma de onda digital binaria de una determinada frecuencia y ciclo de trabajo variable.

PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. En esta técnica se modifica el ciclo de trabajo de una señal pe



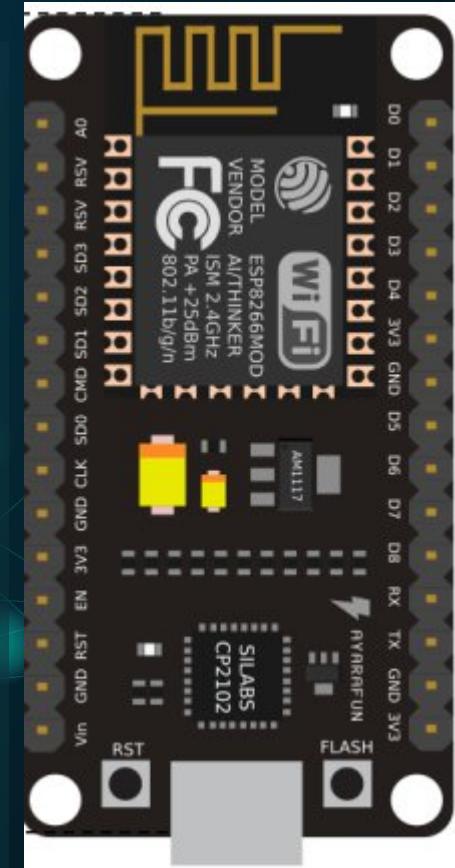
# UNIDAD 4 Interfases de comunicación - IOT

# NodeMCU Placa de desarrollo para IoT con el ESP8266

El objetivo es programar el uC a través de la placa de desarrollo. Lo demás sirve para crear proyectos de manera sencilla.

Incorpora un módulo WiFi que permite crear proyectos del IoT o sistemas inalámbricos.

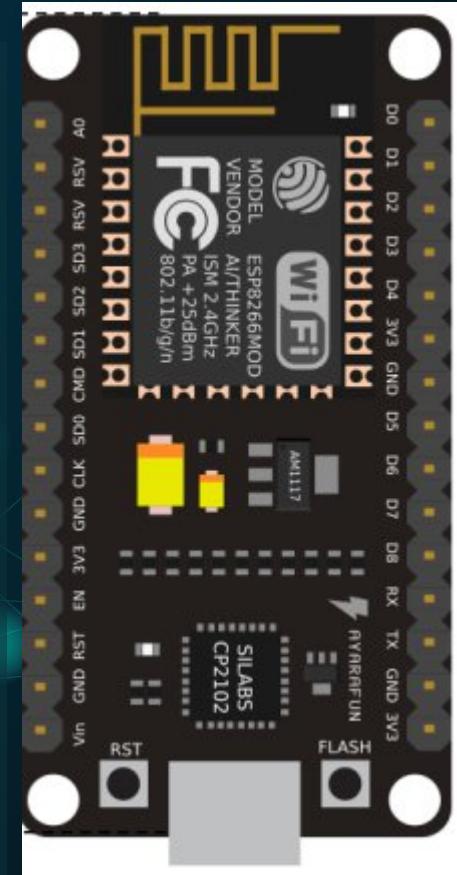
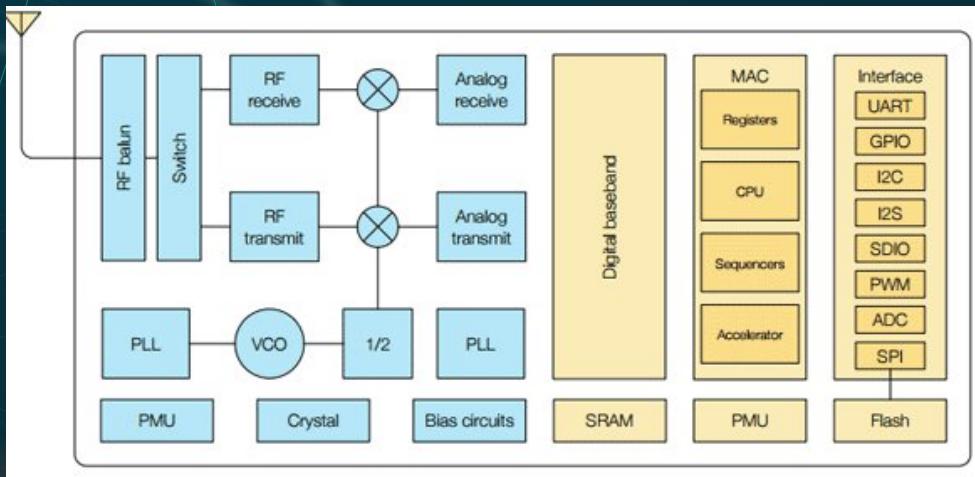
Es el primer paso hacia el IoT. Pueden enviar datos, controlar GPIOs de



# NodeMCU Placa de desarrollo para IoT con el ESP8266

Tensilica L106 de 32-bit

- Arquitectura RISC de 32
- consumo de energía extra-bajo
- velocidad máxima de reloj de 160 MHz,
- Sistema operativo en tiempo real (RTOS)
- Pila Wi-Fi

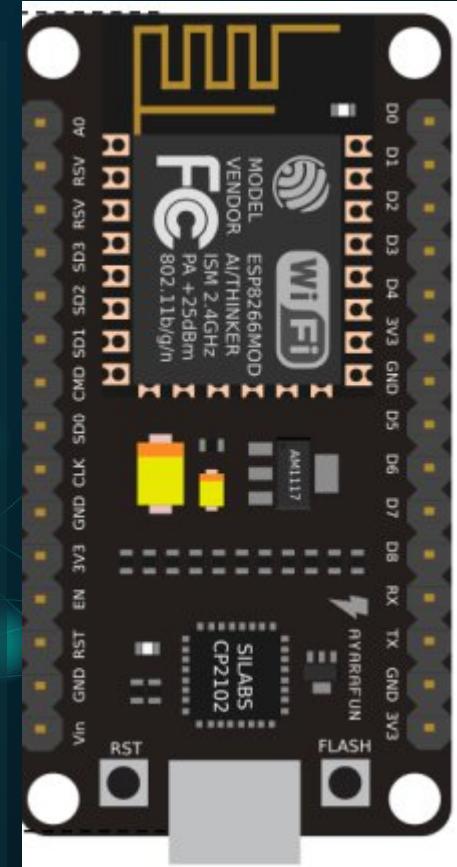


# NodeMCU Placa de desarrollo para IoT con el ESP8266

Es un Sistema en Chip. Es un chip que tiene casi todo integrado para que pueda funcionar como si fuera una computadora. Carece de una memoria para almacenar los programas. Parte de los pines de entrada y salida son utilizados para conectarse a una memoria Flash externa.

Funciones clave de Wi-Fi

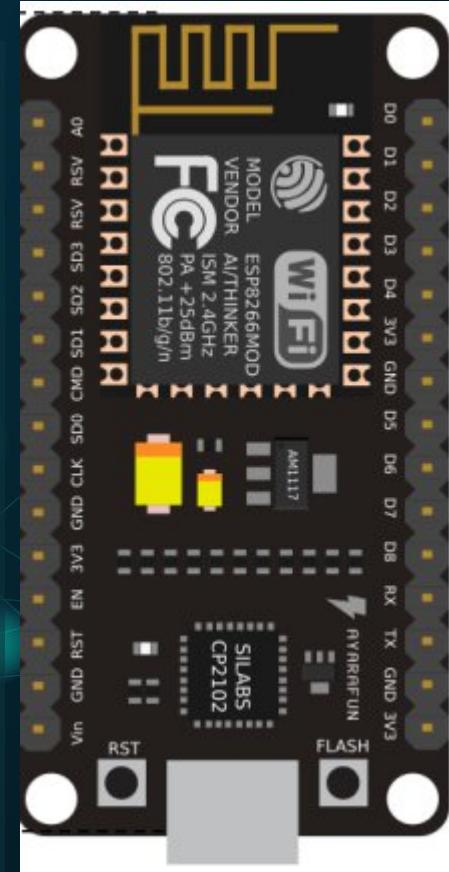
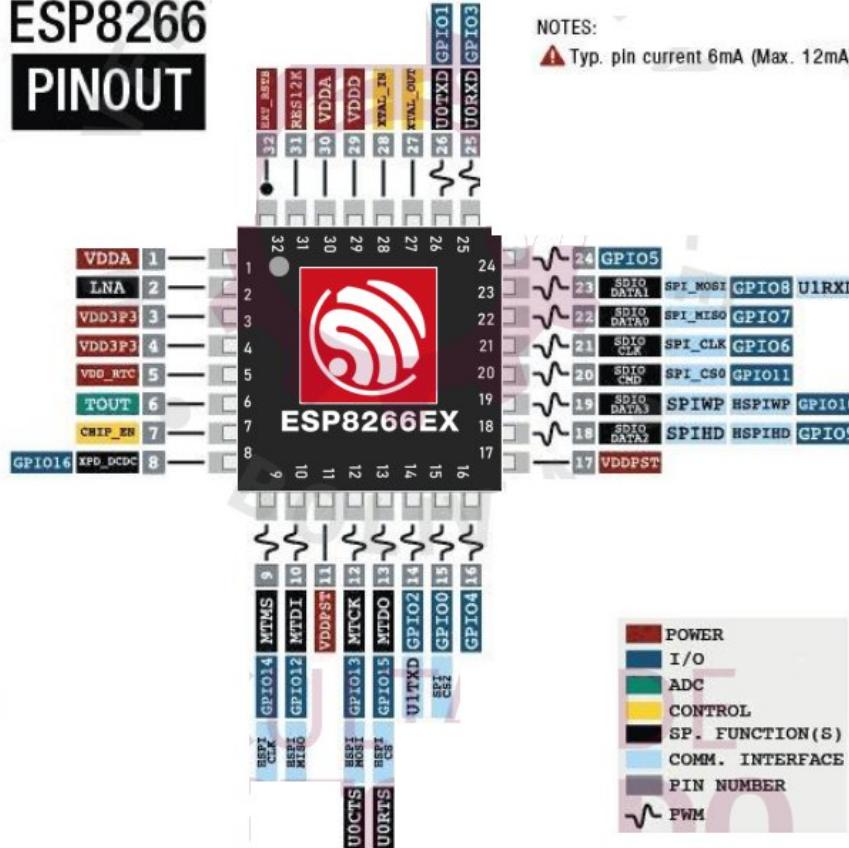
- Compatibilidad con 802.11 b / g / n
- Compatibilidad con 802.11 n (2,4 GHz), hasta 72,2 Mbps
- Desfragmentación • 2 x interfaz Wi-Fi virtual
- Monitoreo automático de balizas (hardware TSF)
- Admite infraestructura de modo estación BSS



# NodeMCU Placa de desarrollo para IoT con el ESP8266

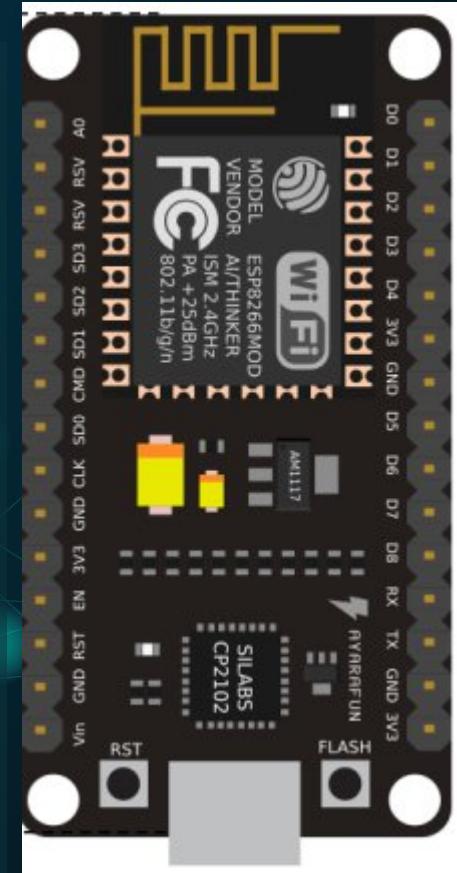
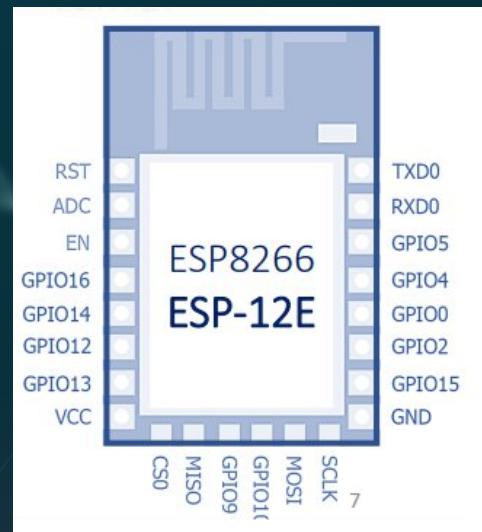
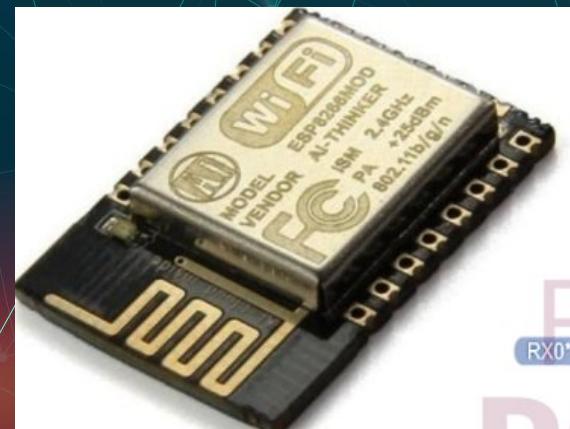


# ESP8266 PINOUT



# NodeMCU Placa de desarrollo para IoT con el ESP8266

- El módulo ESP8266 que utiliza NodeMCU es el ESP-12.
- Dependiendo de la versión de NodeMCU se utilizará módulo ESP-12 y el ESP-12E
- Incorpora memoria Flash y

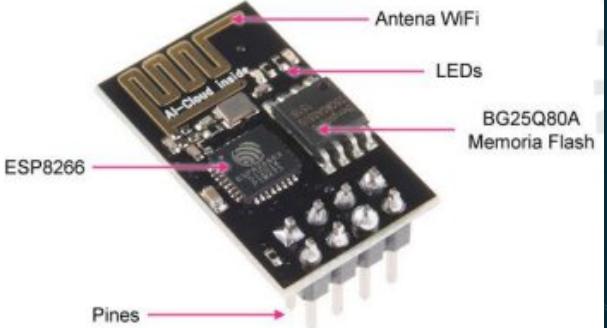


# Módulos ESP8266

Existe una amplia gama de módulos basados en el SoC ESP8266

La diferencia entre ellos es el acceso a los pines. Módulos ESP8266

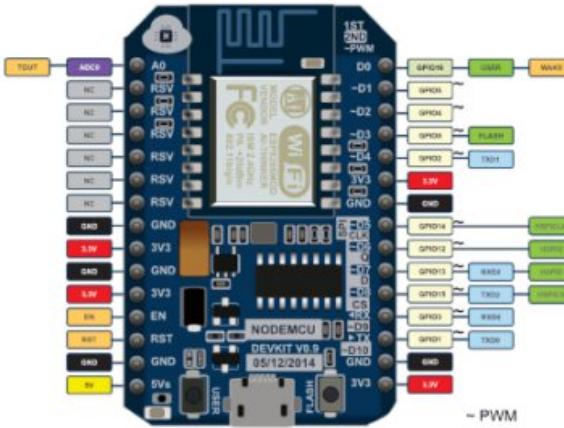
## ESP-01



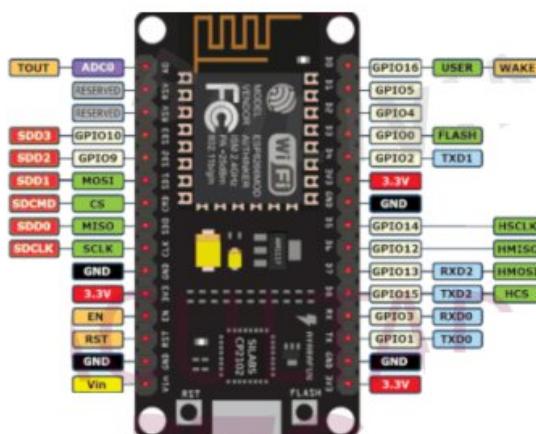
# Versiones de NodeMCU

Debido a que el NodeMCU es una placa de hardware abierto existe varias versiones. Pero todos se basan en los módulos ESP-12 y ESP-12E y el SoC ESP8266.

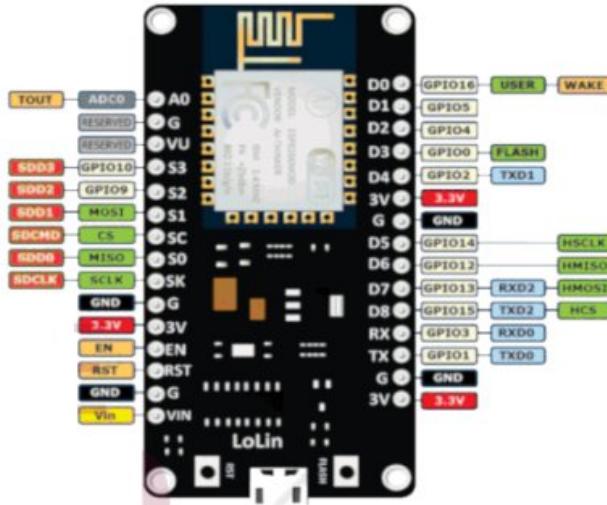
NodeMCU Devkit V0.9  
aka NodeMCU V1



NodeMCU Devkit V1.0  
aka NodeMCU V2



Lolin NodeMCU  
aka "NodeMCU V3"



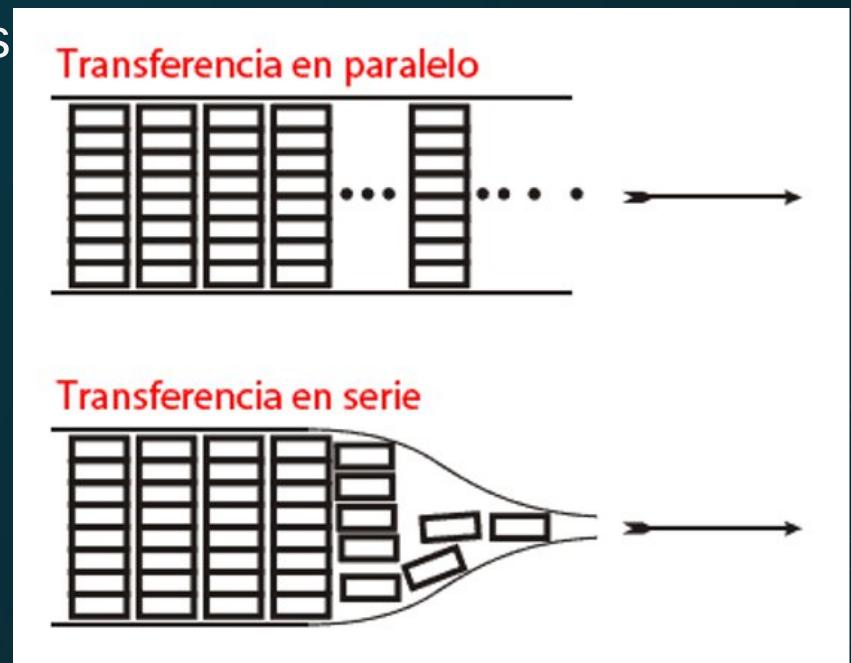
# UNIDAD 4 Interfases de comunicación - IOT

4.1 Protocolos de comunicación

4.2 IoT – Internet de las cosas

# 4.1 Protocolos de comunicación

- Protocolos de comunicación en serie
  - Es una forma de comunicación de datos digitales correspondientes en valores
  - UART
  - I2C
  - SPI



# UART (Universal Asynchronous Receiver Transmitter)

- UART (recepción y transmisión asincrónica universal)
- admite transmisión de datos bidireccional, asíncrona y en serie.
- Tiene dos líneas de datos, una para transmitir (TX) y otra para recibir (RX), las cuales se utilizan para comunicarse a través del pin digital 0, pin digital 1.
- La velocidad de transmisión de datos UART se conoce como tasa BAUD

# UART (Universal Asynchronous Receiver Transmitter)

- el módulo consta de un registro de transmisión y uno de recepción para almacenar los datos. Fiel a su naturaleza asíncrona, la transmisión y la recepción en un nodo son impulsadas por su generador de reloj local.

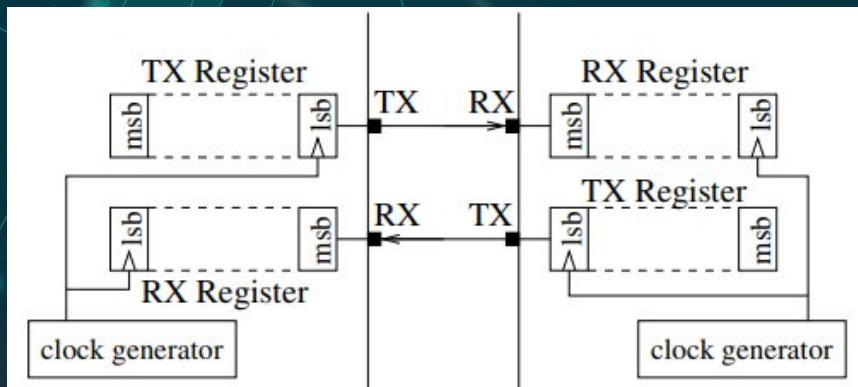
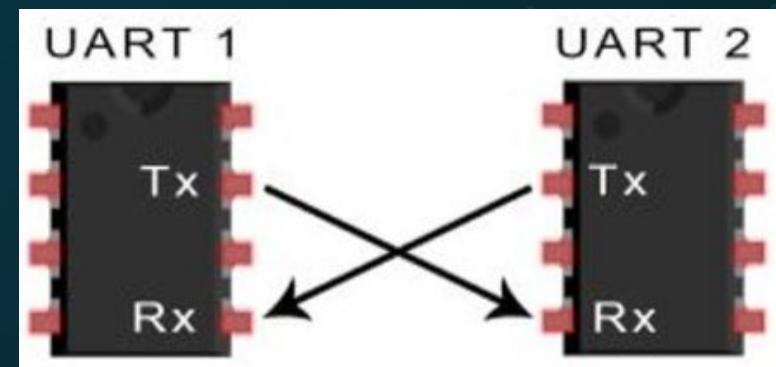
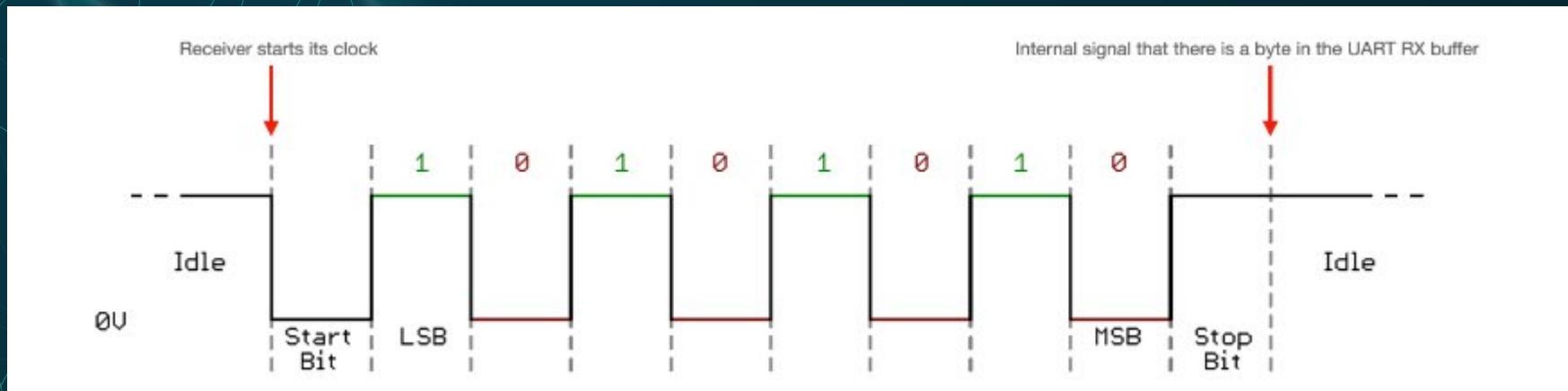


Figure 3.1: Basic structure of a UART module.



# UART (Universal Asynchronous Receiver Transmitter)

- UART (recepción y transmisión asincrónica universal)



# UART (Universal Asynchronous Receiver

## Transmitter)

- Simple de operar, bien documentado ya que es un método ampliamente utilizado con muchos recursos en línea.
- No se necesita reloj
- Bit de paridad para permitir la verificación de errores

## Desventajas:

- El tamaño de la trama de datos está limitado a solo 9 bits
- No se pueden usar múltiples sistemas maestros y esclavos
- Las tasas de baudios de cada UART deben estar dentro del 10 % entre sí para evitar la pérdida de datos.
- Bajas velocidades de transmisión de datos

# I2C (Inter-Integrated Circuit)

- ❑ Es un protocolo de comunicaciones en serie similar a UART.
- ❑ Es un bus síncrono que opera según un principio maestro-esclavo.
- ❑ Es un bus serial síncrono bidireccional simple de dos cables.
- ❑ El protocolo incluye mecanismos de arbitraje de bus y, por lo tanto, permite la coexistencia de varios maestros.
- ❑ Tiene 2 líneas que son SCL (línea de reloj en serie) y SDA (puerto de aceptación de línea de datos en serie)

# I2C (Inter-Integrated Circuit)

- Tiene 2 líneas que son SCL (línea de reloj en serie) y SDA (puerto de aceptación de línea de datos en serie)

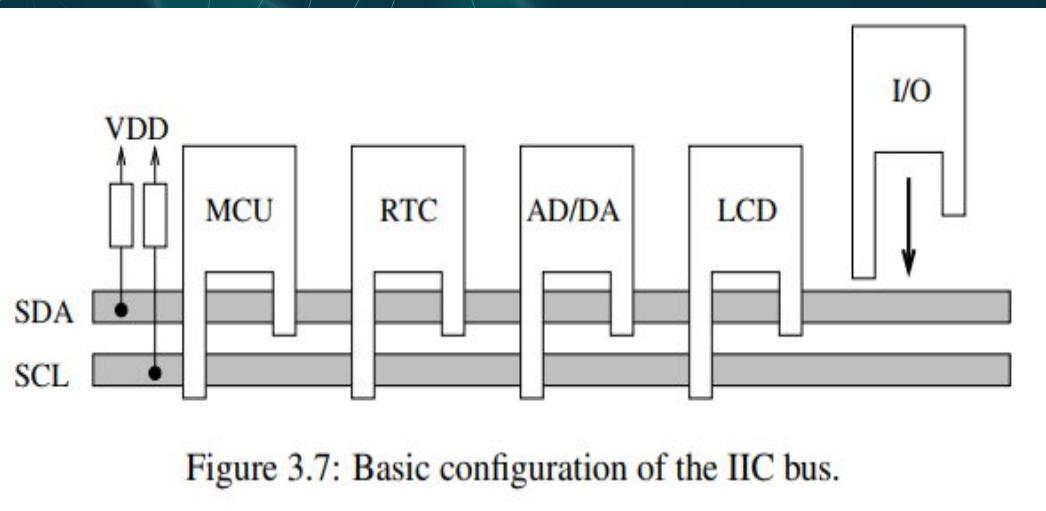
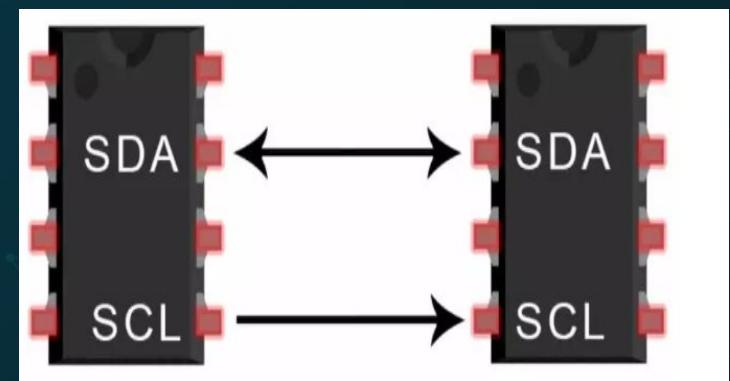
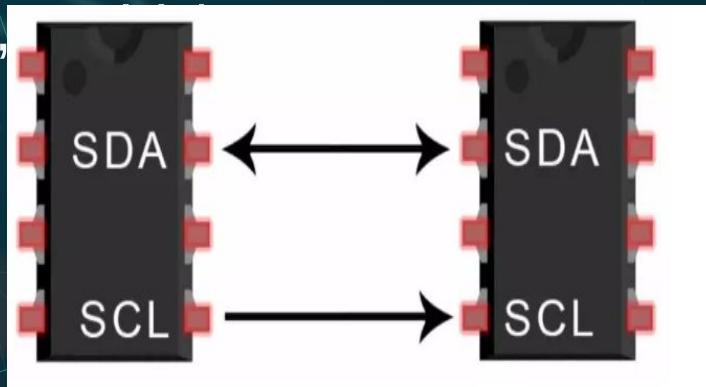


Figure 3.7: Basic configuration of the IIC bus.



# I2C (Inter-Integrated Circuit)

- Tiene tres modos de velocidad:
- el modo estándar abarca velocidades de transmisión de hasta 100 kbit/s.
- el modo rápido amplía este rango a 400 kbit/s
- el modo de alta velocidad aumenta la velocidad de transmisión a 3,



# I2C (Inter-Integrated Circuit)

## Ventajas:

- Flexible, ya que admite comunicación multimaestro y multiesclavo.
- Simple ya que solo utiliza 2 cables bidireccionales
- Adaptable ya que puede adaptarse a las necesidades de varios dispositivos esclavos.
- Soporta múltiples maestros.

## Desventajas:

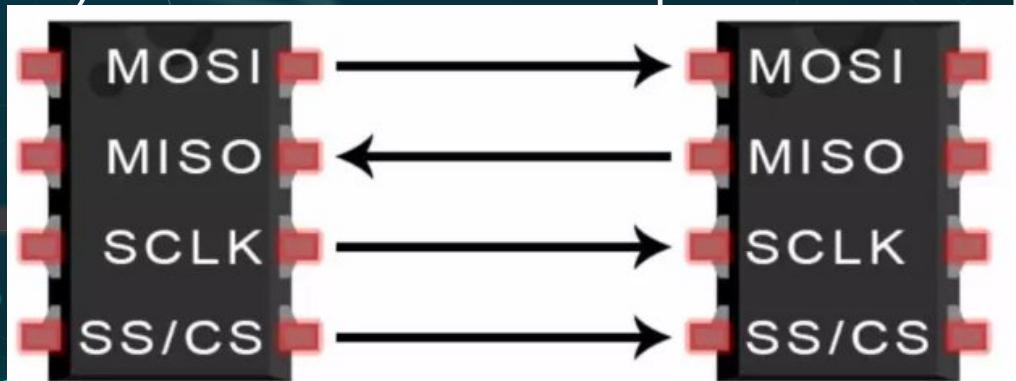
- Velocidad más lenta
- Puede volverse complejo a medida que aumenta la cantidad de dispositivos.

# SPI (Serial port interface)

- ❑ Es similar a I2C y es una forma diferente de protocolo de comunicaciones en serie especialmente diseñado para que se conecten los MCU.
- ❑ Funciona en dúplex completo donde los datos se pueden enviar y recibir simultáneamente.
- ❑ Se utiliza en lugares donde la velocidad es importante. (por ejemplo, tarjetas SD, módulos de visualización,etc)

# SPI (Serial port interface)

- MOSI (Master Out, Slave In): Esta línea es utilizada por el maestro para transmitir datos al esclavo.
- MISO (Master In, Slave Out): Esta línea es utilizada por el esclavo para transmitir datos al maestro.
- SCK (System Clock): Esta línea es utilizada por el maestro para transmitir la señal del reloj.
- SS (Slave Select): Esta línea es utilizada por el maestro para seleccionar un esclavo.



# SPI (Serial port interface)

## Ventajas:

- no es un sistema complicado en direccionamiento como I2C.
- Es el protocolo más rápido en comparación con UART e I2C.
- Sin bits de inicio y parada a diferencia de UART
- Líneas MISO y MOSI separadas

## Desventajas:

- Hay más puertos Pin ocupados
- No hay un control de flujo especificado y ningún mecanismo de reconocimiento confirma si los datos se reciben a diferencia de I2C
- Utiliza cuatro líneas: MOSI, MISO, NCLK, NSS