

# Electrónica Digital y Microcontroladores

## Tema 5.1: Introducción a los microcontroladores

---

Josué Meneses Díaz

[josue.meneses@usach.cl](mailto:josue.meneses@usach.cl)

Universidad de Santiago de Chile

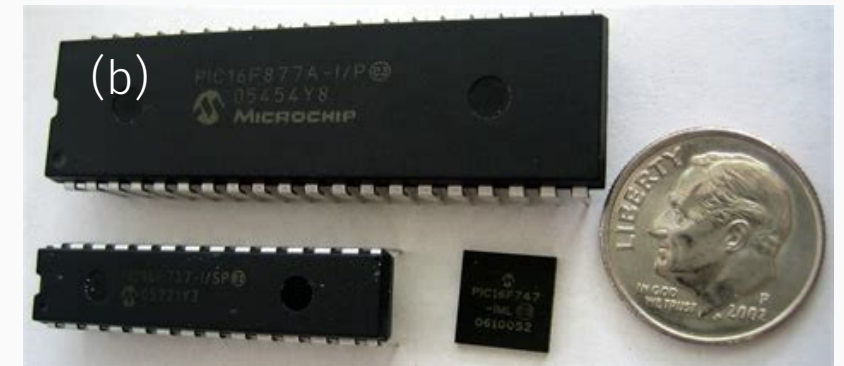
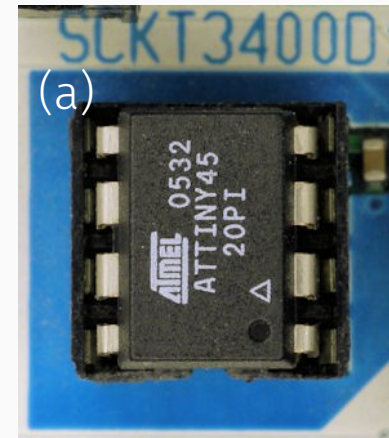
31-07-2024

# Objetivos

- Introducción a los Microcontroladores (MCUs).
  - Partes internas.
  - Filosofía de funcionamiento.
- Entender que es Arduino
- Introducir el lenguaje de programación Arduino
  - Laboratorio 5 – “Cronómetro simple”

# ¿Qué es un Microcontrolador?

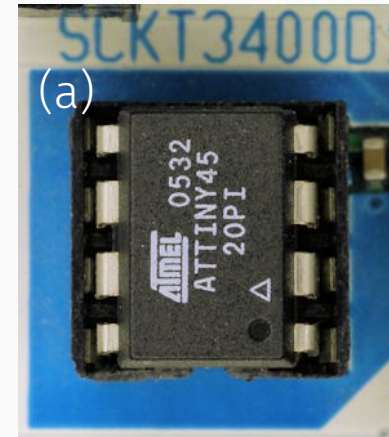
- El microcontrolador o uC o  $\mu C$  o MCU es un circuito integrado (CI) programable.
- Computador dedicado a una tarea específica.
  - La tarea es definida por el programador.
- Permite:
  - Automatización de procesos.
  - Digitalización de medidas.
  - Control de sistemas.
- En un pequeño espacio es capaz de realizar las mismas tareas que un circuito digital
  - Abaratamiento de costos.
  - Disminución de espacio.
  - Rapidez de desarrollo.



(a) Ejemplo de MCU de AVR. (b) Ejemplo de PIC de Microchip

# ¿Qué es un Microcontrolador?

- Un MCU cumple con:
  - Comunicación con el exterior.
  - Tamaño reducido
  - Bajo costo
  - Protección de código.



(a) Ejemplo de MCU de AVR. (b) Ejemplo de PIC de Microchip

# Idea General

Se quiere diseñar una FSM que cumpla con:

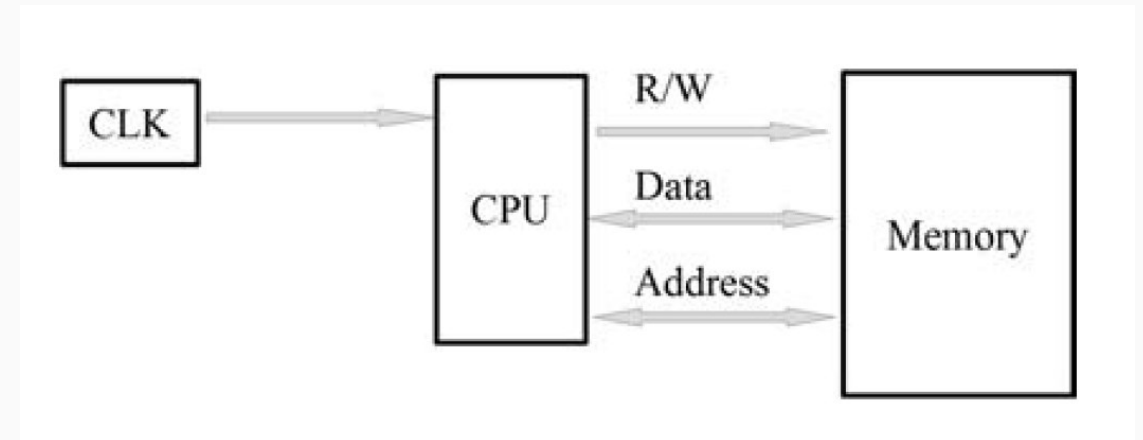
- Contador que recorra de 0 a  $n$
- Las cuentas son utilizadas para recorrer un set de instrucciones

(a) 

(b) 

# Esquema básico de una MCU.

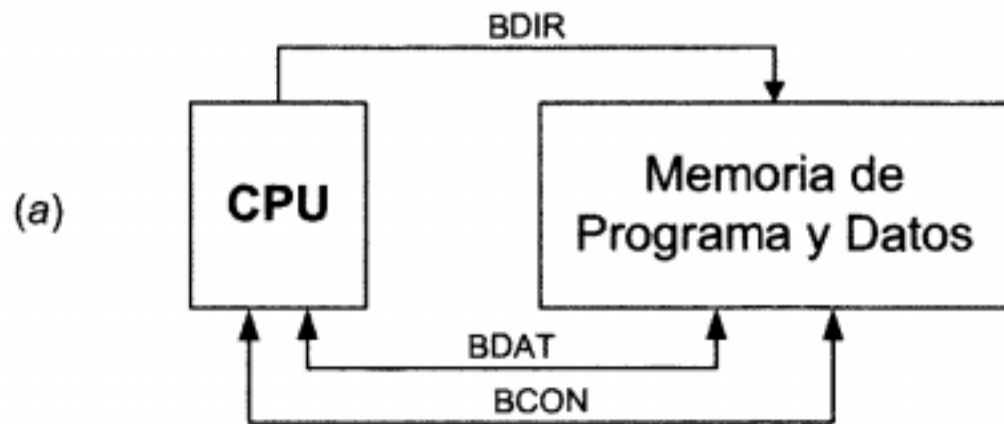
- Internamente, un MCU puede ser dividido en tres secciones
- CPU (central processing Unit).
  - ALU
- Memoria.
- Entradas y salidas.



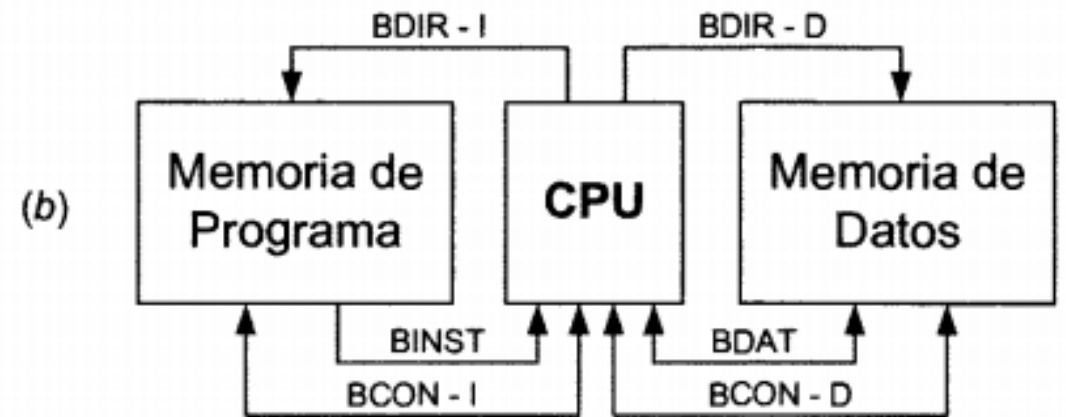
Grace, Thomas. *Programming and Interfacing Atmel AVR Microcontrollers*. Cengage Learning PTR, 2016.

# Arquitecturas

## Von Neumann



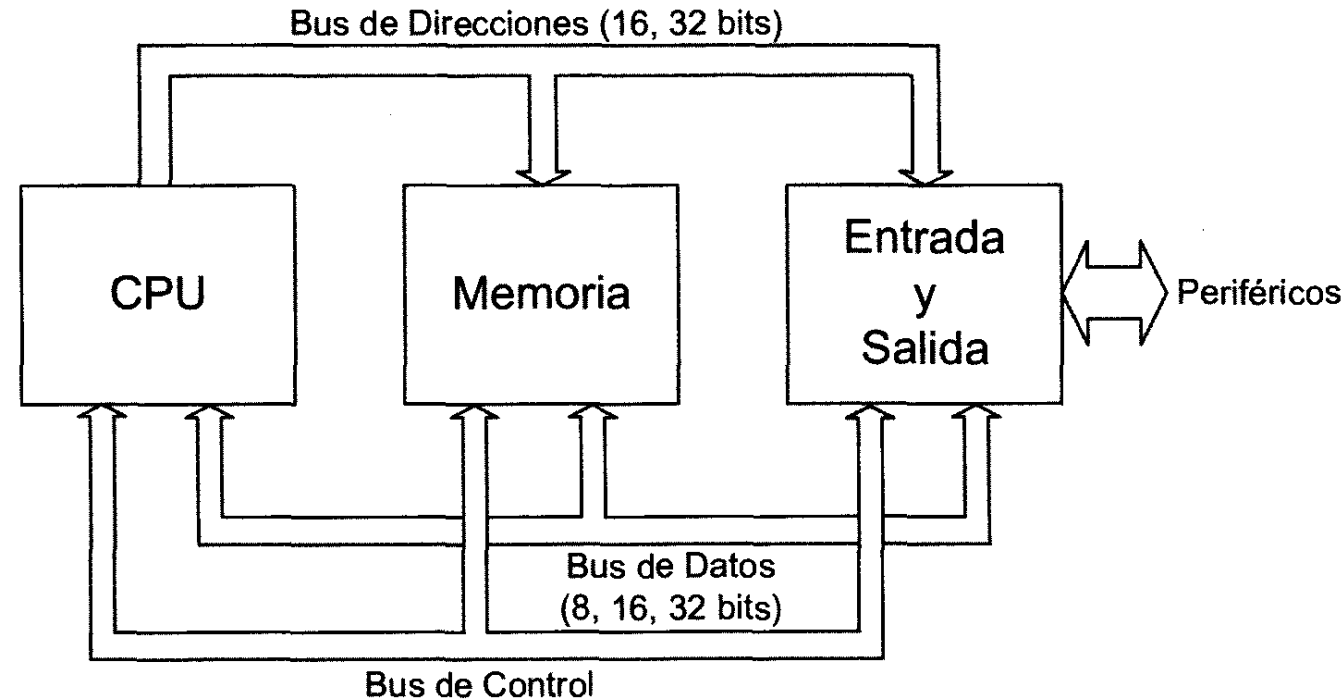
## Harvard



Valdés F, Areny RP. Microcontroladores fundamentos y aplicaciones con PIC. vol. 1149. Marcombo; 2007.

# Esquema básico de una MCU.

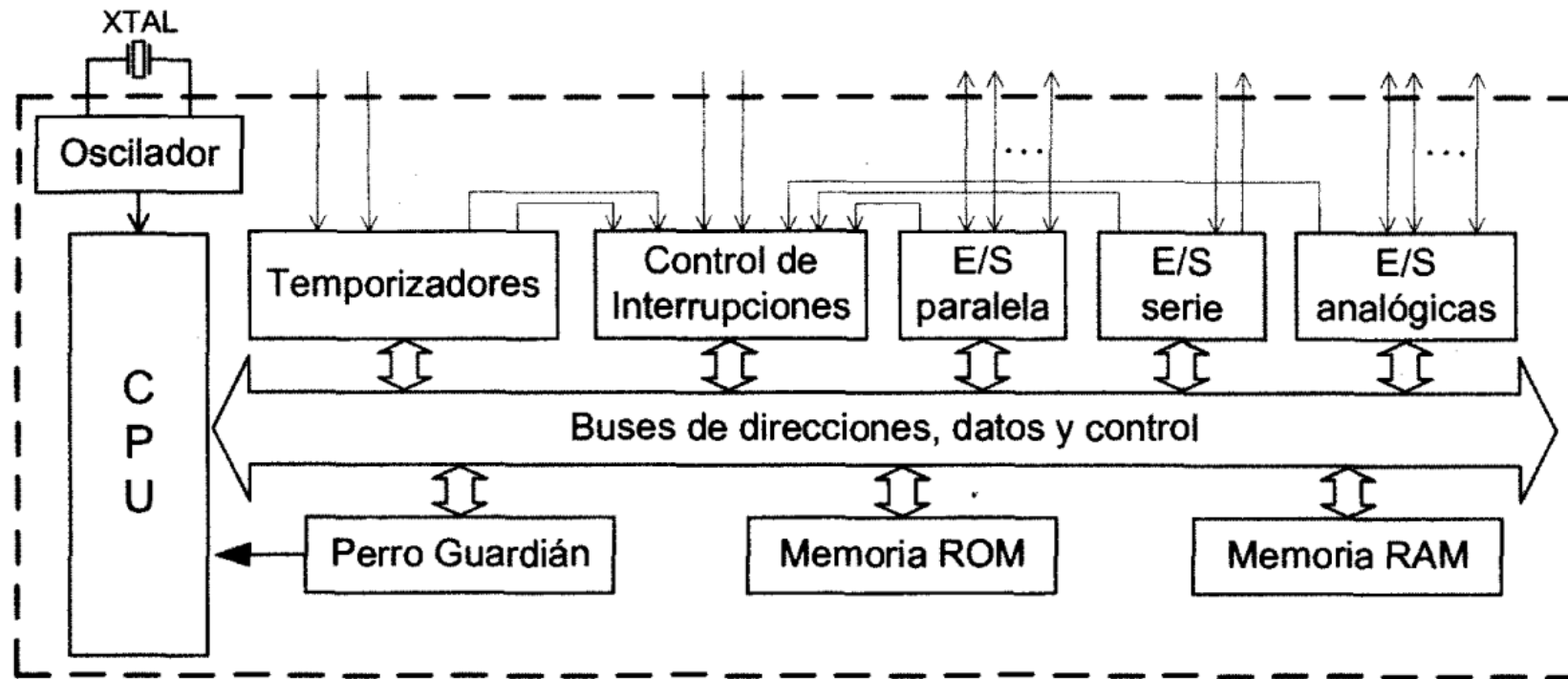
- Internamente, un MCU puede ser dividido en tres secciones
  - - CPU (central processing Unit).
    - ALU
  - - Memoria.
  - - Entradas y salidas.



Valdés F, Areny RP. Microcontroladores fundamentos y aplicaciones con PIC. vol. 1149. Marcombo; 2007.



# Esquema básico de una MCU.

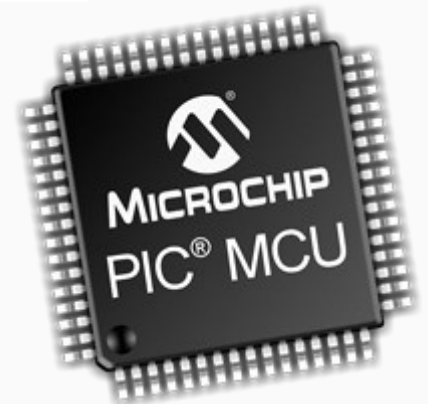


*Figura 1.2 Esquema de bloques general de un microcontrolador.*

Valdés F, Areny RP. Microcontroladores fundamentos y aplicaciones con PIC. vol. 1149. Marcombo; 2007.

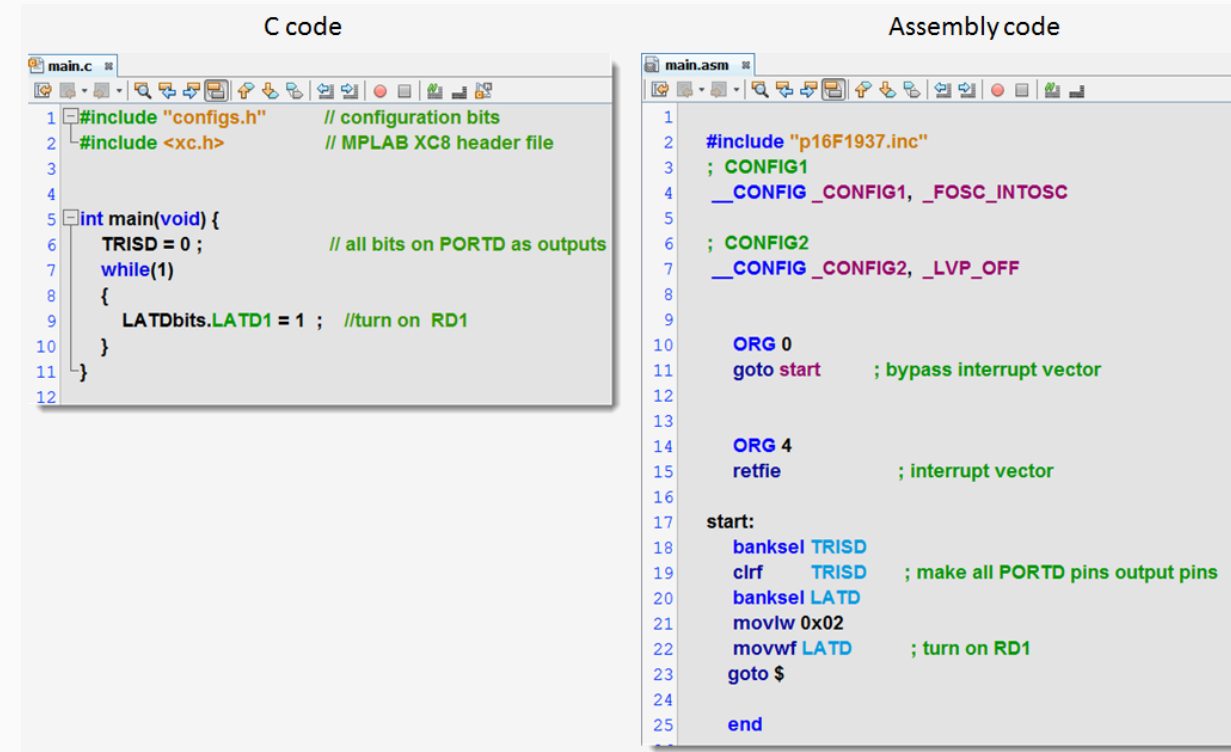
# Tipos de MCU

- Existen distintos tipos de microcontroladores en el mercado, que dependerán del fabricante y arquitectura de construcción.
  - PIC - Microchip
  - AVR - Microchip
  - MSP430 – TI
  - Etc.



# Lenguaje de programación

- Cuando queremos programar un MCU necesitamos generar un archivo **.hex** que será grabado al MCU.
  - Basic.
  - C.
  - Ensamblador.
- El archivo es compilado
- PIC
  - XC8.
  - CSS compiler
  - Mikro C
- Arduino
  - Arduno IDE
  - Microchip Studio



The image displays two code editors side-by-side, illustrating the compilation process from C to Assembly for a PIC microcontroller.

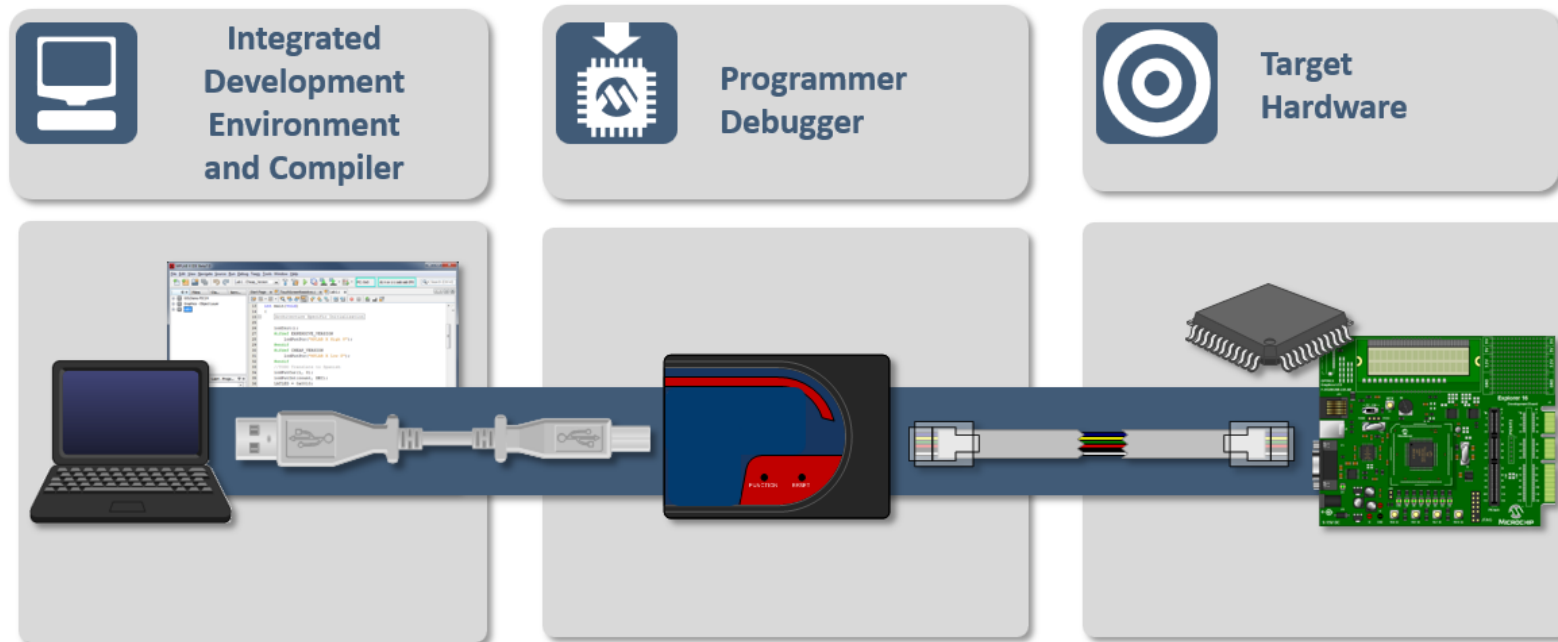
**C code (main.c):**

```
1 #include "configs.h" // configuration bits
2 #include <xc.h>      // MPLAB XC8 header file
3
4
5 int main(void) {
6     TRISD = 0;        // all bits on PORTD as outputs
7     while(1)
8     {
9         LATDbits.LATD1 = 1; //turn on RD1
10    }
11 }
12
```

**Assembly code (main.asm):**

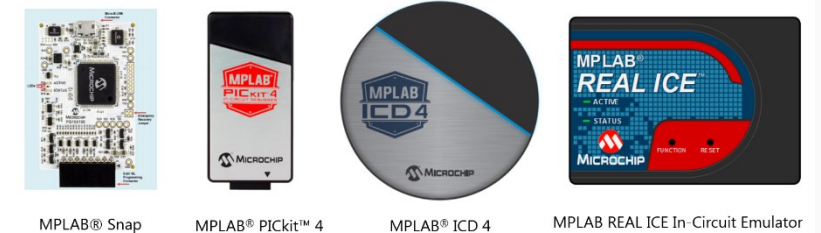
```
1
2 #include "p16F1937.inc"
3 ; CONFIG1
4 __CONFIG __CONFIG1, _FOSC_INTOSC
5
6 ; CONFIG2
7 __CONFIG __CONFIG2, _LVP_OFF
8
9
10 ORG 0
11 goto start ; bypass interrupt vector
12
13
14 ORG 4
15 retfie ; interrupt vector
16
17 start:
18 banksel TRISD
19 clrf TRISD ; make all PORTD pins output pins
20 banksel LATD
21 movlw 0x02
22 movwf LATD ; turn on RD1
23 goto $
24
25 end
```

# Proceso de Diseño - PIC



IDE: MPLAB X  
Compilador :XC8

Pickit 2, 3, 4

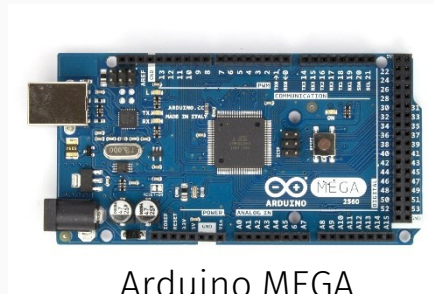


Developer Help Training n.d.  
<https://developerhelpttraining.thinkific.com/courses/take/introtomplabx/texts/8040431-ide-overview>

# Proceso de Diseño - Arduino



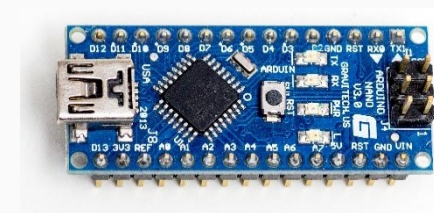
IDE Arduino



Arduino MEGA

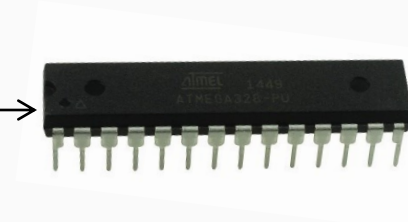


Arduino UNO



Arduino Nano

Vcc + GND



Atmega 328a

Oscilador

# ARDUINO

# ¿Qué es Arduino?

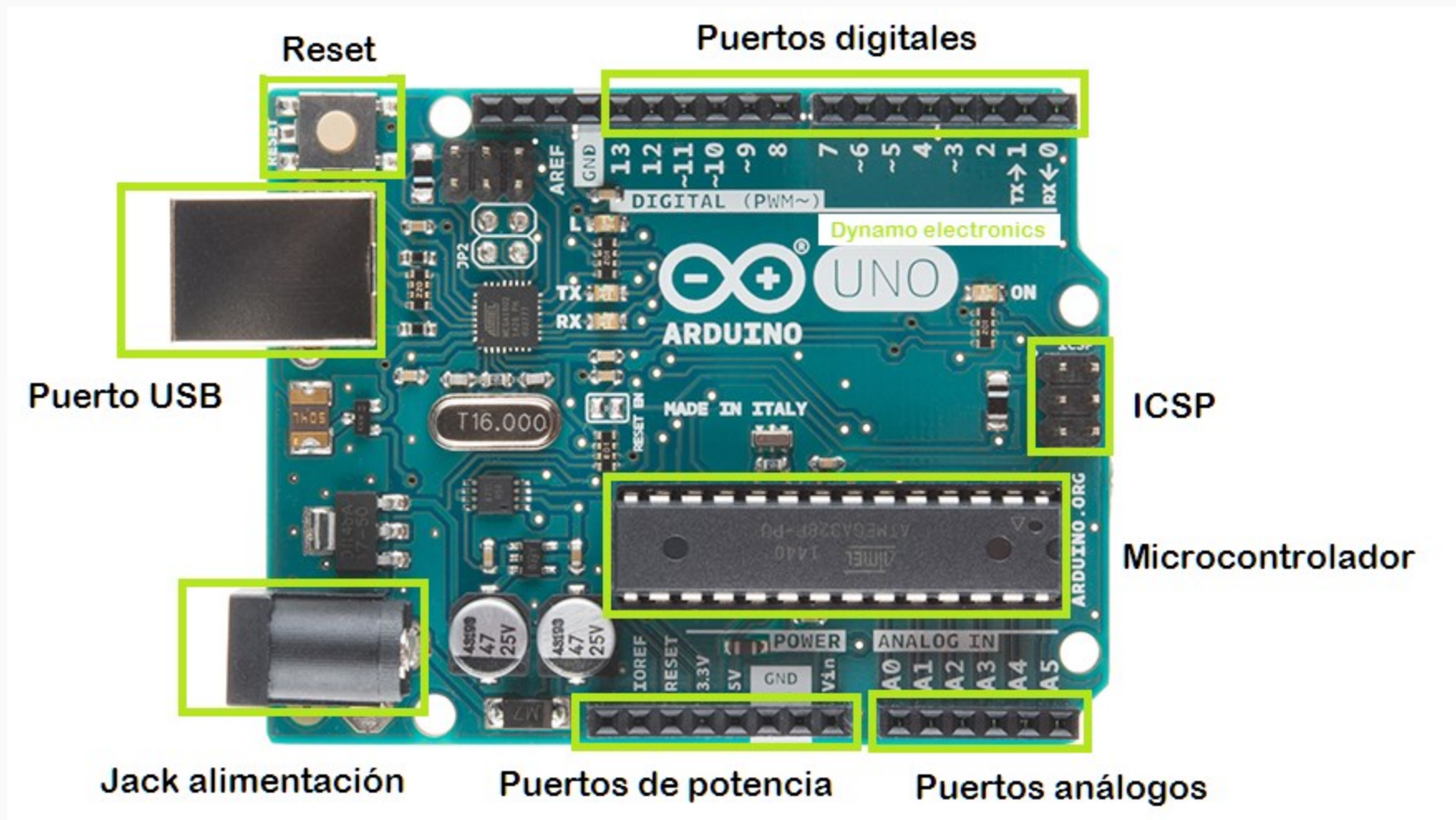
Arduino son en realidad tres cosas:

- Una **placa hardware libre** que incorpora un *microcontrolador Atmega328* reprogramable y una serie de pines que permite conectar periféricos. (sensores, actuadores, etc.).
- Un **software** o “entorno de desarrollo” (IDE) gratis, libre y multiplataforma
  - Basado en [Processing](#)
- Un **lenguaje de programación** libre.
  - Basado en C y C++



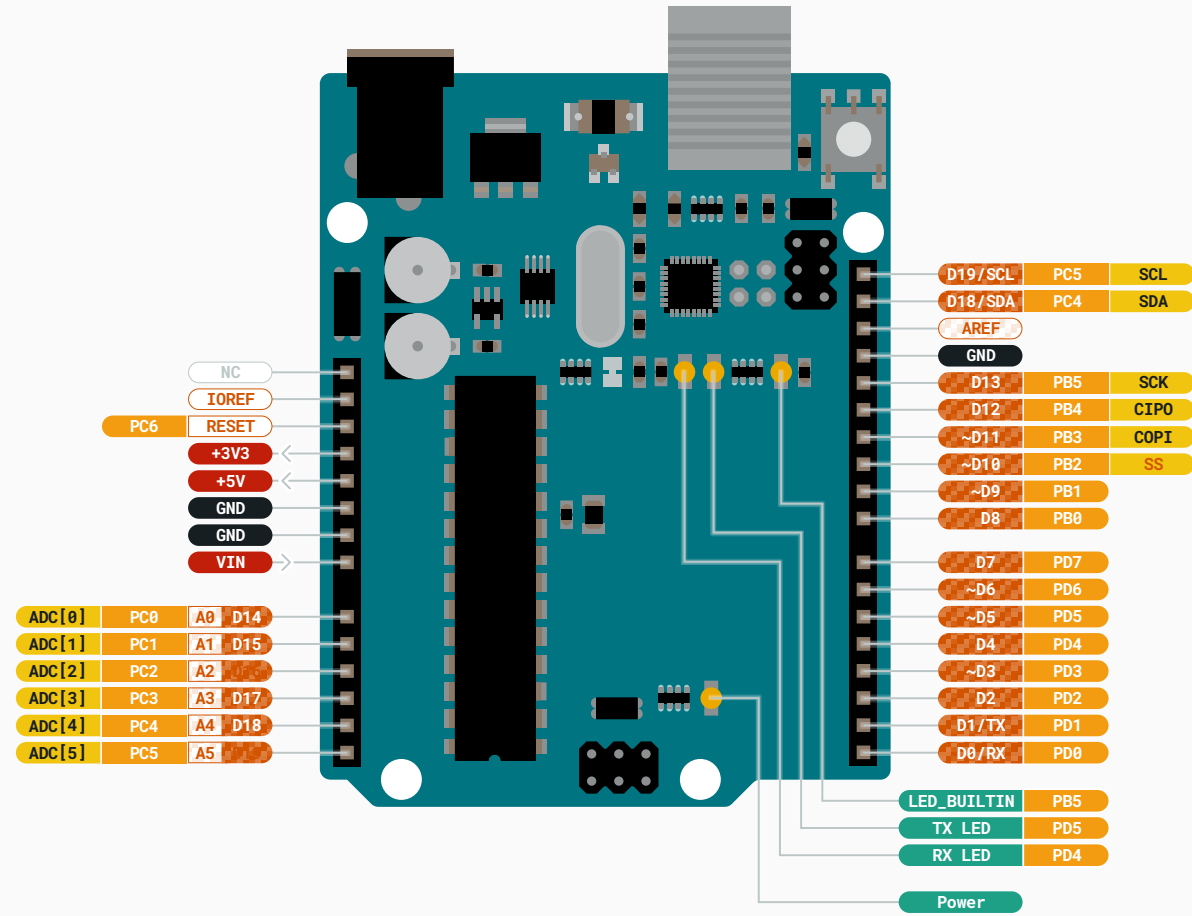
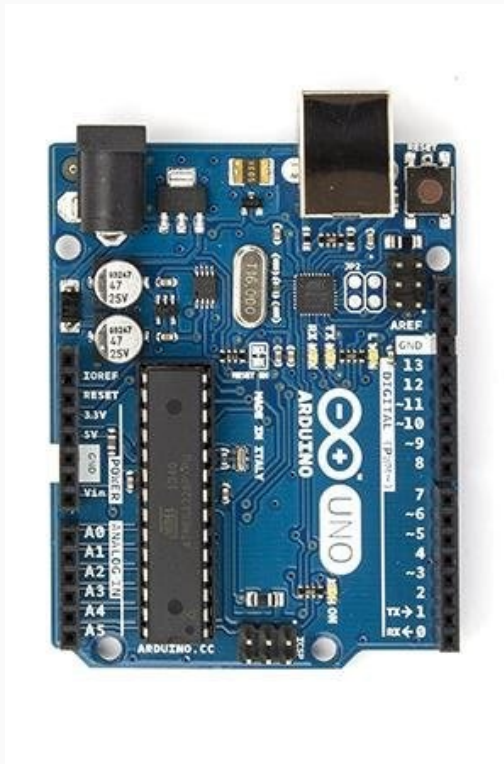


# Vista general Arduino UNO



Partes Arduino UNO – [Freddy Huamachuco](#)





Ground

Power

LED

Internal Pin

SWD Pin

Digital Pin

Analog Pin

Other Pin

Microcontroller's Port

Default

⚠️ **MAXIMUM** current per I/O pin is 20mA

⚠️ **MAXIMUM** current per +3.3V pin is 50mA

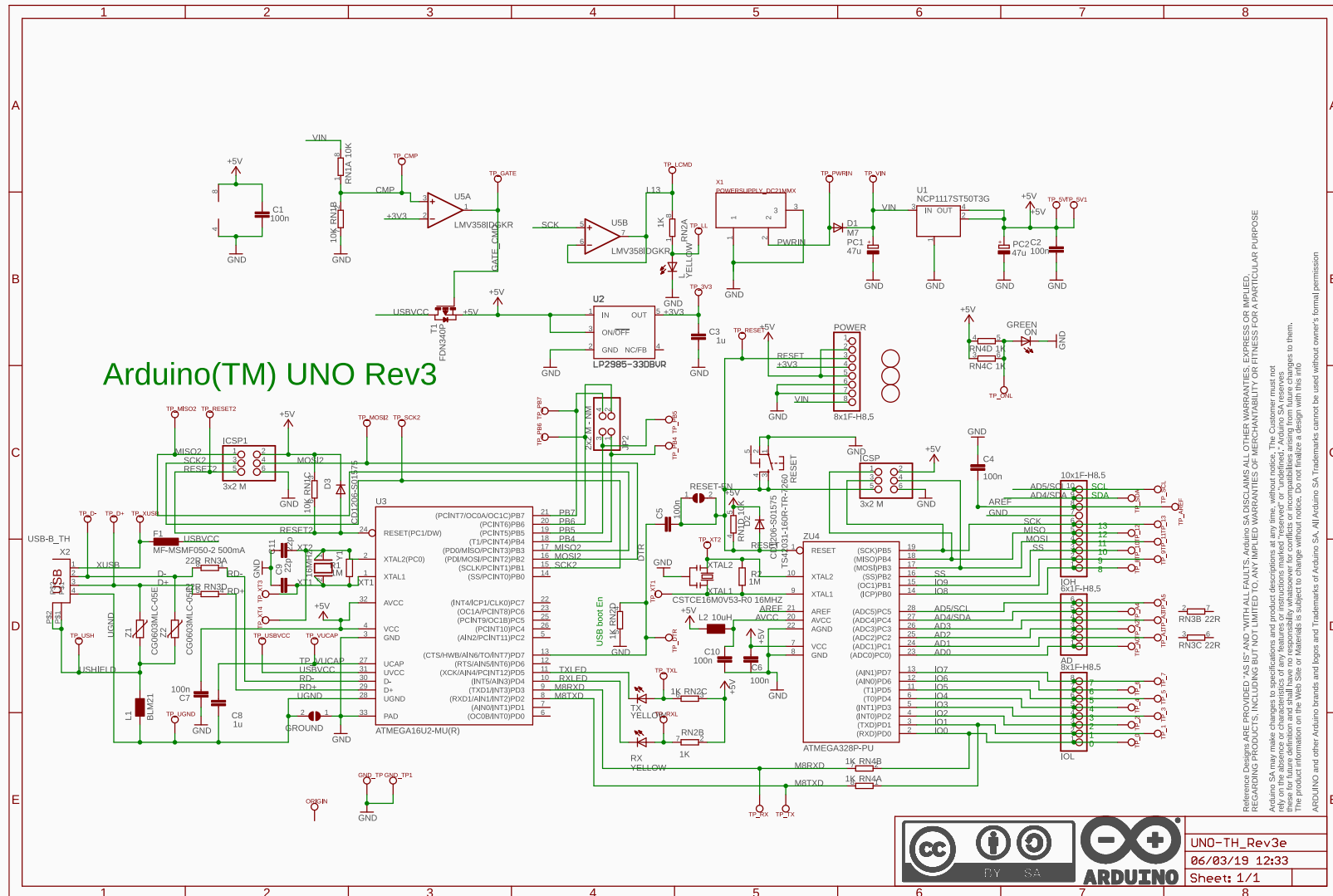
**VIN** 6-20 V input to the board.

NOTE: CIPO/COPI have previously been referred to as MISO/MOSI

Pinos de Salida/Entrada - [Arduino](#)

7/31/2024

19



Esquemático Arduino UNO - Arduino

# LENGUAJE DE PROGRAMACIÓN ARDUINO

# Estructura básica de un programa

```
void setup() {  
    // put your setup code here  
}  
void loop() {  
    /* put your main code here,  
    to run repeatedly:*/  
}
```

```
void setup() {}
```

Código de configuración. Ejecutado una sola vez.

```
void loop() {
```

Loop principal. Se repite infinitamente!!!

```
// Es utilizado para comentarios de una línea.
```

```
/* */ Comentarios multilínea.
```

En realidad estas dos partes no son más que dos funciones

```
tipo nombre_funcion( variables ){}  
  
void setup() {}  
void loop() {}
```

# Variables

Siempre tienen que ser declara, especificando, aunque sea una vez, su tipo.

Cada línea tiene que terminar en ; (salvo las funciones).

```
int x; // Variable `x` declarada
```

```
int n = 0; // Variable `n` declarada y  
asignada
```

```
void    // The void keyword is used only in function declarations. It indicates that the
function is expected to return no information
byte    // A byte stores an 8-bits unsigned number, from 0 to 255 (2^8-1).
int      // 16-bits (2-byte). Range of -32,768 to 32,767 (-2^15, 0, 2^15-1).
unsigned int // 0 to 65,535 ((2^16) - 1).
long     // 32-bits (4 bytes), from -2,147,483,648 to 2,147,483,647.
unsigned long // 0 to 4,294,967,295 (2^32 - 1).
float    // -3.4028235E+38 to 3.4028235E+38 . They are stored as 32 bits (4 bytes).
double   // Idem. to float on Arduino Uno. On Arduino Due, 8-byte (64 bit) precision.
char     // A data type used to store a character value.
char myChar = 'A';
```

```
String
String stringOne = "Hello String";
```

### Variable Declaration - Arduino.cc

# Diagramas de flujo

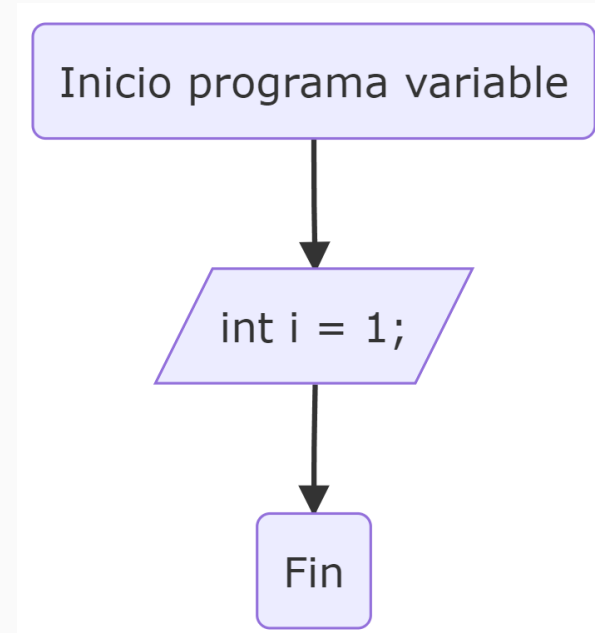
- Es una representación gráfica del programa que estamos desarrollando.
- Permite explicar de una forma visual las decisiones tomadas por el programador.
- Se recomienda crear inicialmente un diagrama de flujo antes de escribir el programa.

Símbolo inicio/fin

Inicio/Fin del programa

Símbolo entrada

int i = 1;



# Condicional If else

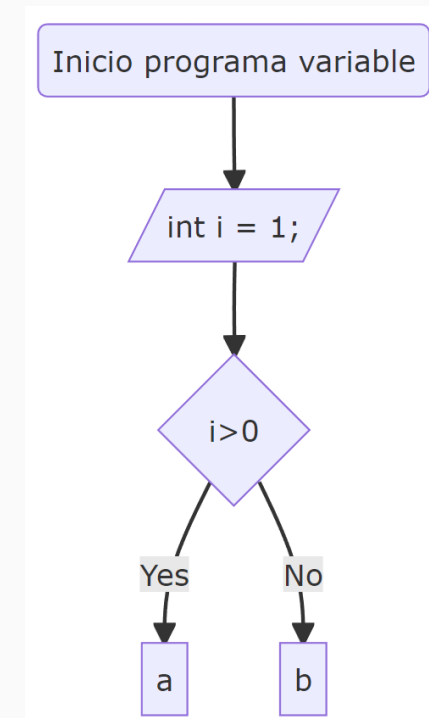
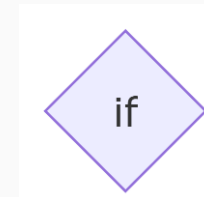
- La instrucción if busca una condición y ejecuta la siguiente instrucción o conjunto de declaraciones si la condición es 'verdadera'.

## Sintaxis

```
if (condition) {  
    //statement(s)  
}
```

```
x == y (x is equal to y)  
x != y (x is not equal to y)  
x < y (x is less than y)  
x > y (x is greater than y)  
x <= y (x is less than or equal to y)  
x >= y (x is greater than or equal to y)
```

## Símbolo





# Condicional If else

- La instrucción if busca una condición y ejecuta la siguiente instrucción o conjunto de declaraciones si la condición es 'verdadera'.

## Sintáxis

```
if (condition) {  
    //statement(s)  
}  
  
x == y (x is equal to y)  
x != y (x is not equal to y)  
x < y (x is less than y)  
x > y (x is greater than y)  
x <= y (x is less than or equal to y)  
x >= y (x is greater than or equal to y)
```

## Ejemplo

```
if (condition1) {  
    // do Thing A  
}  
else if (condition2) {  
    // do Thing B  
}  
else {  
    // do Thing C  
}
```

If - Arduino  
else - Arduino

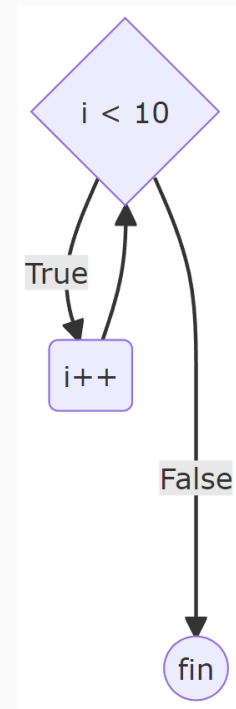
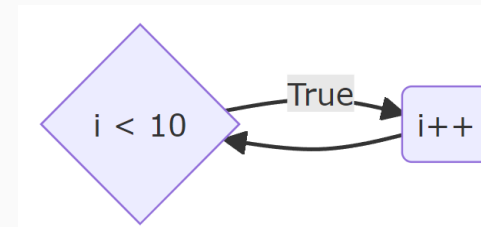
# Ciclo For

- La instrucción **for** se usa para repetir un bloque de declaraciones entre llaves. Por lo general, se usa un contador de incrementos.

## Sintáxis

```
for (initialization; condition;  
    increment) {  
    // statement(s);  
}
```

## Símbolo



for - Arduino

# Ciclo For

- La instrucción **for** se usa para repetir un bloque de declaraciones entre llaves. Por lo general, se usa un contador de incrementos.

## Sintaxis

```
for (initialization; condition;  
    increment) {  
    // statement(s);  
}
```

## Ejemplo

```
for (int i = 0; i <= 255; i++) {  
    delay(10);  
}
```

[for - Arduino](#)

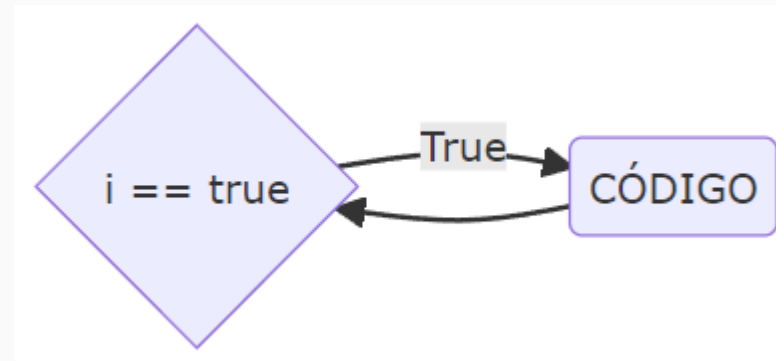
# Ciclo While

- Un ciclo **while** se repetirá de forma continua e infinita, hasta que la expresión dentro del paréntesis () se vuelva falsa.

## Sintáxis

```
while (condition) {  
    // statement(s)  
}
```

## Símbolo



# Ciclo While

- Un ciclo **while** se repetirá de forma continua e infinita, hasta que la expresión dentro del paréntesis () se vuelva falsa.

## Sintáxis

```
while (condition) {  
    // statement(s)  
}
```

## Ejemplo

```
int var = 0;  
while (var < 200) {  
    // do something repetitive 200 times  
    var++;  
}
```

[while - Arduino](#)

# Funciones especiales - delay()

Pausa el programa durante el tiempo (en milisegundos) especificado como parámetro.

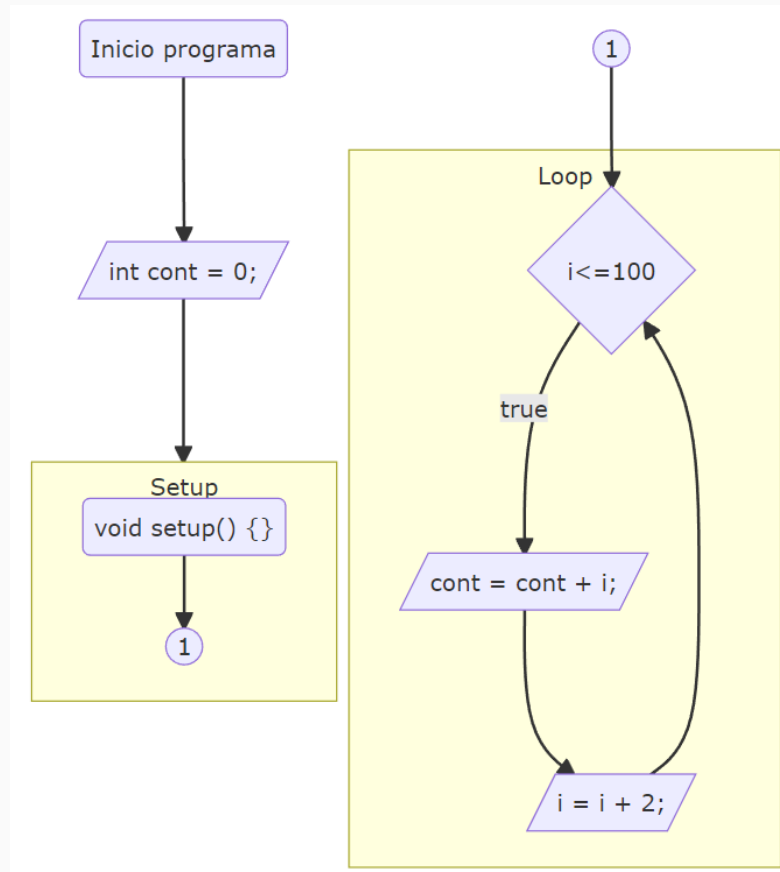
## Sintaxis

```
delay(ms); // = ms*delayMicroseconds()
```

Función delay()

# Ejemplo 1

Crear un programa que sume los números impares del 1 a 100 utilizando el lenguaje de Arduino.



[Editor online](#)

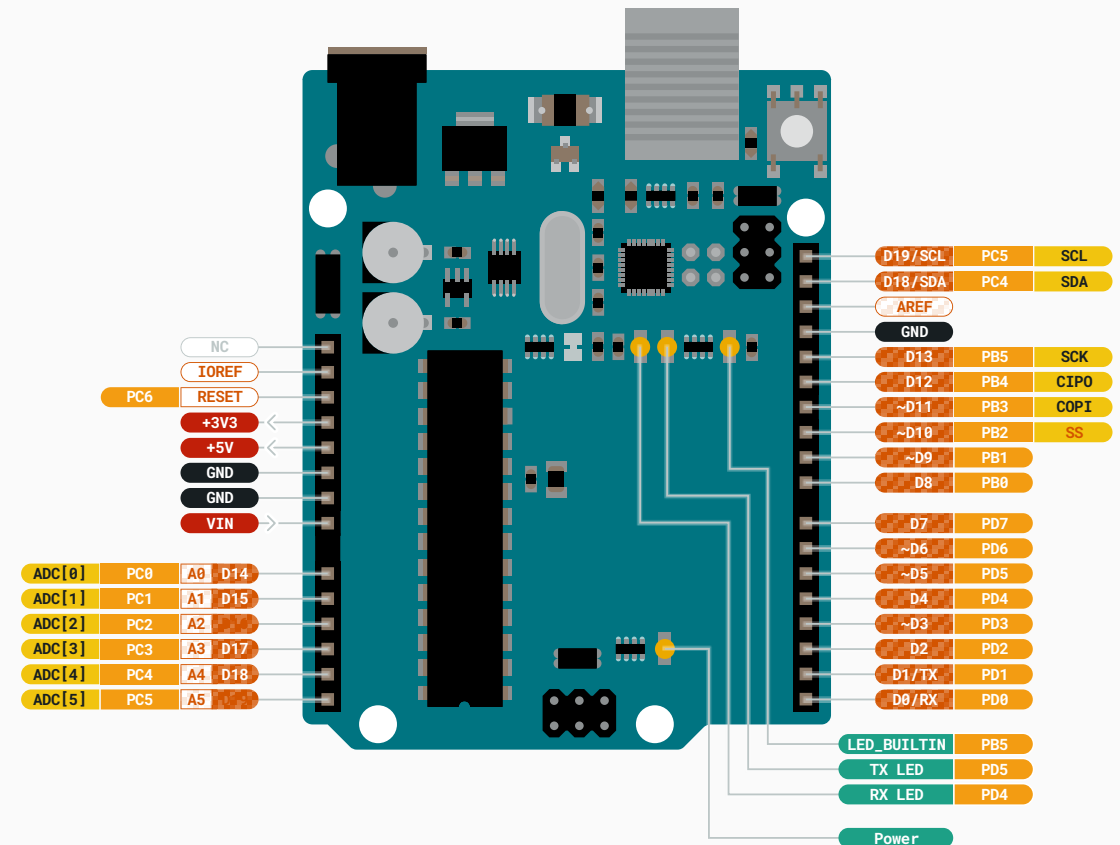
```
int cont = 0;

void setup() {
}

void loop() {
  for( int i = 1; i<=100; i=i+2){
    cont = cont + i;
  }
}
```

# Pines Digitales

- Una de las características más importantes de un MCU es su *capacidad de comunicarse con el exterior*.
- Configuración de periféricos
- Comunicación entre equipos
- Para ellos utilizamos los **pines digitales**:
- Pueden ser configurados como entradas y salidas.
- Los reconocemos con el nombre de *Dxx* donde *xx* es un número.
- *\*Solo pueden tomar valores Lógicos (1 y 0).*



Ground  
Power  
LED  
Internal Pin  
SWD Pin

Digital Pin  
Analog Pin  
Other Pin  
Microcontroller's Port  
Default

**MAXIMUM** current per I/O pin is 20mA  
**MAXIMUM** current per +3.3V pin is 50mA

**VIN** 6-20 V input to the board.  
NOTE: CIP0/COPI have previously been referred to as MISO/MOSI



# pinMode()

`pinMode()` Configura el comportamiento de un pin digital del Arduino.

## Sintáxis

```
pinMode(pin, modo)
```

pin: n° pin de Arduino  
modo: INPUT o OUTPUT

## Ejemplo

```
void setup(){  
    // D0 como pin de salida  
    pinMode(0, OUTPUT);  
}
```

# digitalWrite() y digitalRead()

## digitalWrite()

Permite *enviar* **valores lógicos** por los pines digitales de una Tarjeta Arduino previa configuración mediante

### Sintaxis

```
digitalWrite(PIN_DIGITAL, ESTADO);
```

**ESTADO:** HIGH, LOW

## digitalRead()

Permite *recibir* **valores lógicos** por los pines digitales de una Tarjeta Arduino previa configuración mediante

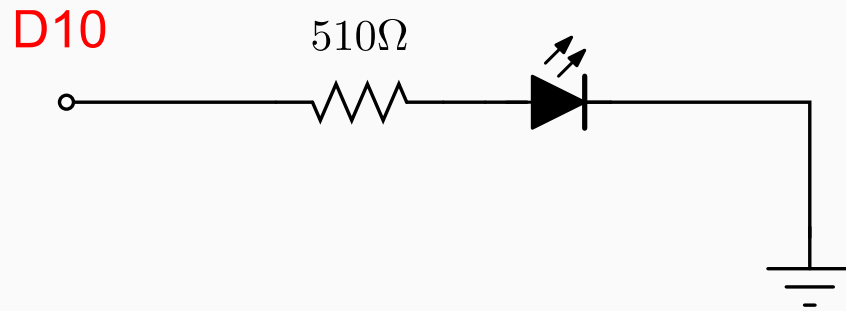
### Sintaxis

```
digitalRead(PIN_DIGITAL);
```

[digitalWrite - Arduino](#)  
[digitalRead - Arduino](#)

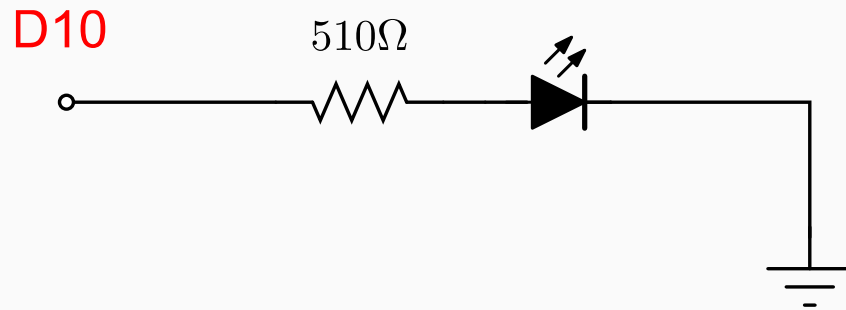
# Ejemplo 2 – Encendido de un diodo LED

Escriba un programa que encienda y apague un diodo LED cada 1s que se encuentra conectado al pin digital 10.

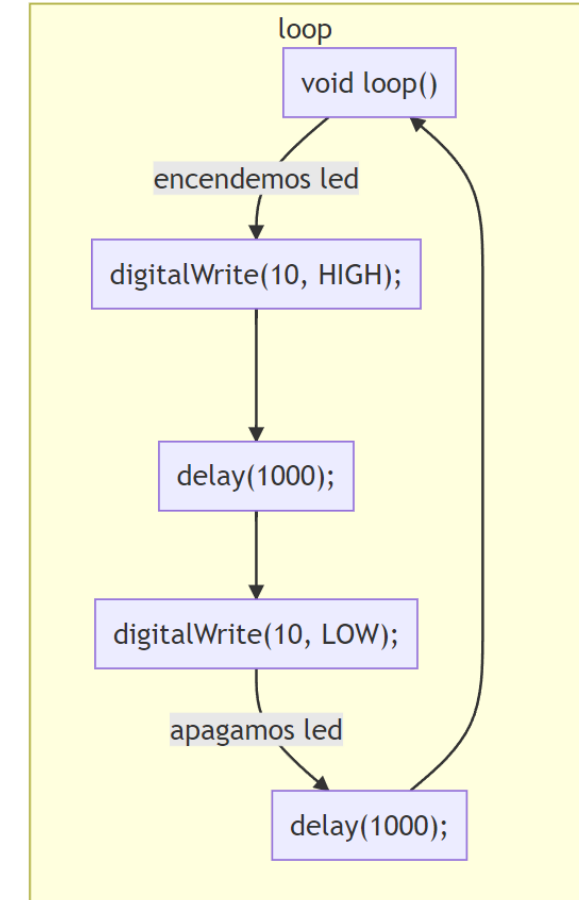
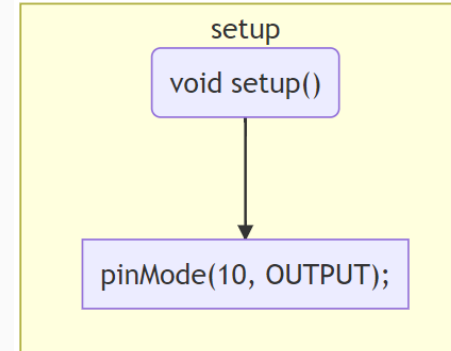


# Ejemplo 2 – Encendido de un diodo LED

Escriba un programa que encienda y apague un diodo LED cada 1s que se encuentra conectado al pin digital 10.



```
void setup(){  
  pinMode(10, OUTPUT);  
}  
  
void loop(){  
  // Encendemos D0  
  digitalWrite(10, HIGH);  
  delay(1000);  
  // Apagamos D0  
  digitalWrite(10, LOW);  
  delay(1000);  
}
```



# Funciones en Arduino (c/c++)

- Los bloques usados hasta el momento no son más que funciones en c. en general podemos crear las funciones que necesitemos para simplificar nuestros programas

## Sintáxis

```
tipo nombre_funcion(tipo var1, tipo var2, ...){  
    // contenido de la función  
    return variable_tipo_definida;  
}
```

## Ejemplo

```
void loop(){  
    int a = 10;  
    int b = 20;  
}  
  
int sum(int x, int y){  
    return x+y;  
}
```

# Ejemplo 3

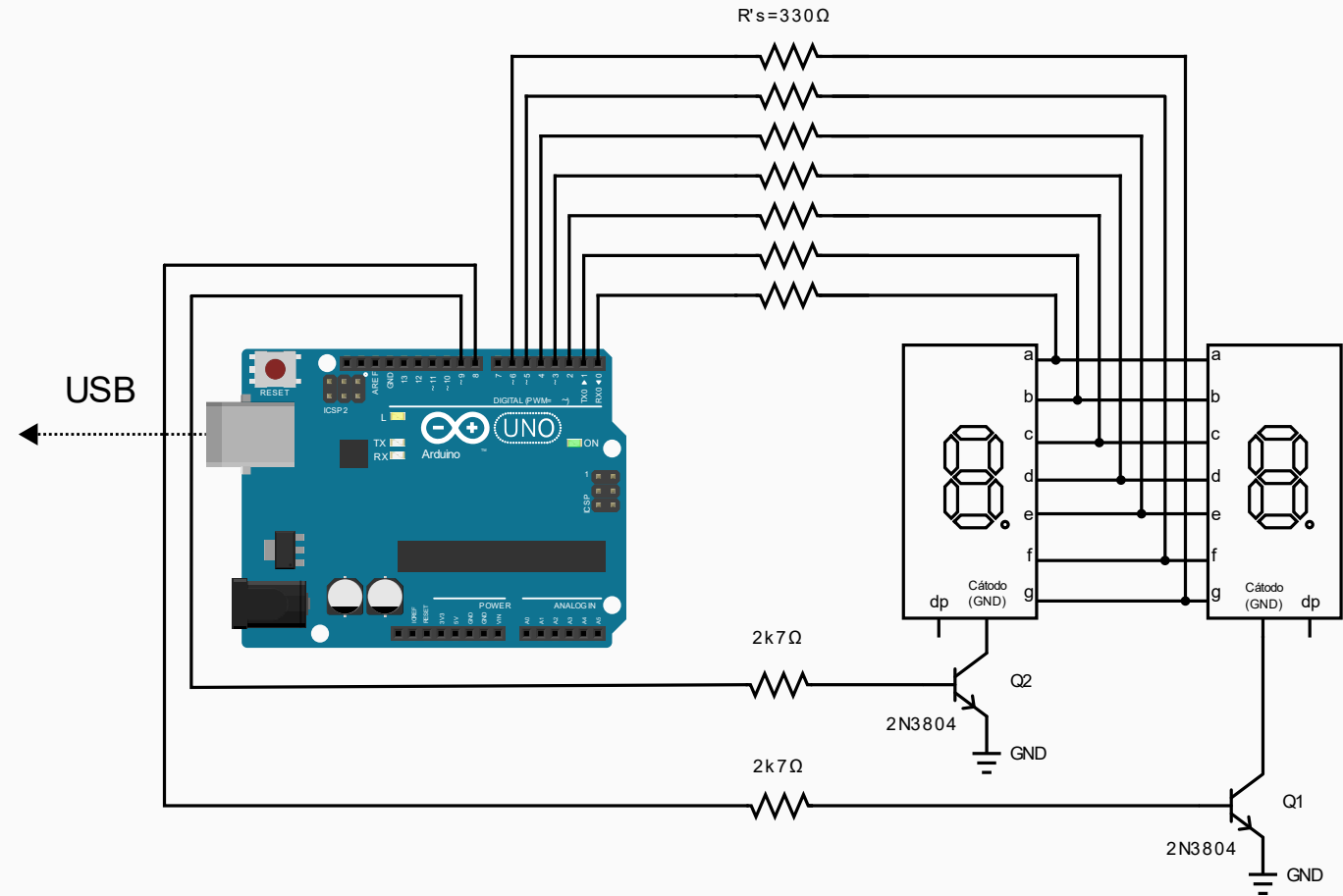
- Reescriba el programa de encendido del diodo LED ahora utilizando funciones.

```
void setup(){  
    pinMode(0, OUTPUT);  
}  
void loop(){  
    encender_led(2000);  
    apagar_led(1000);  
}
```

```
void encender_led(int t){  
    digitalWrite(0, HIGH);  
    delay(t);  
}  
void apagar_led(int t){  
    digitalWrite(0, LOW);  
    delay(t);  
}
```

# Laboratorio 5 – “Cronómetro simple”

- Controlaremos dos D7S con los pines digitales de un Arduino.
- Se utilizarán dos transistores NPN como conmutadores que activan/desactivan los D7S.
- Los D7S cambiarán cada segundo hasta 60s.



# Resumen

- Un MCU es un CI programable.
- Existen una gran cantidad de MCU y compañías que lo fabrican. PIC y Arduino son ejemplos de MCU fabricado por Microchip.
- Existen varios lenguajes de programación de un MCU, entre ellos C, que se ha convertido en un estándar para varios MCU.
- El proceso de diseño usando Arduino es:
  - IDE
  - Generación de .hex
  - Carga del programa



# Referencias

- Valdés, Fernando, and Ramón Pallás Areny. 2007. *Microcontroladores Fundamentos y Aplicaciones Con PIC*. Vol. 1149. Marcombo.
- Grace, Thomas. *Programming and Interfacing Atmel AVR Microcontrollers*. Cengage Learning PTR, 2016.
- Arduino. «Arduino Reference - Arduino Reference». Accedido 16 de junio de 2022. <https://www.arduino.cc/reference/en/>.
- Mermaidjs. «Online FlowChart & Diagrams Editor - Mermaid Live Editor». Accedido 16 de junio de 2022. <https://mermaid.live>.
- Artero, Óscar Torrente. *ARDUINO. Curso práctico de formación*. RC libros, 2013.
- Dunbar, Norman. *Arduino Software Internals: A Complete Guide to How Your Arduino Language and Hardware Work Together*. Apress, 2020.
- Microchip. «ATmega328 | Microchip Technology». Accedido 16 de junio de 2022. <https://www.microchip.com/en-us/product/ATmega328>.