

Sistemas embebidos

Clase de Práctico N°8

Procesamiento Digital de Señales



INGENIERÍA
BIOLÓGICA

Contenidos

01

Arquitectura de un arduino

Central y periféricos

02

Características de cada placa

Uno, Mega, Due

03

Análisis de un programa

Puerto serie, frecuencia de muestreo

04

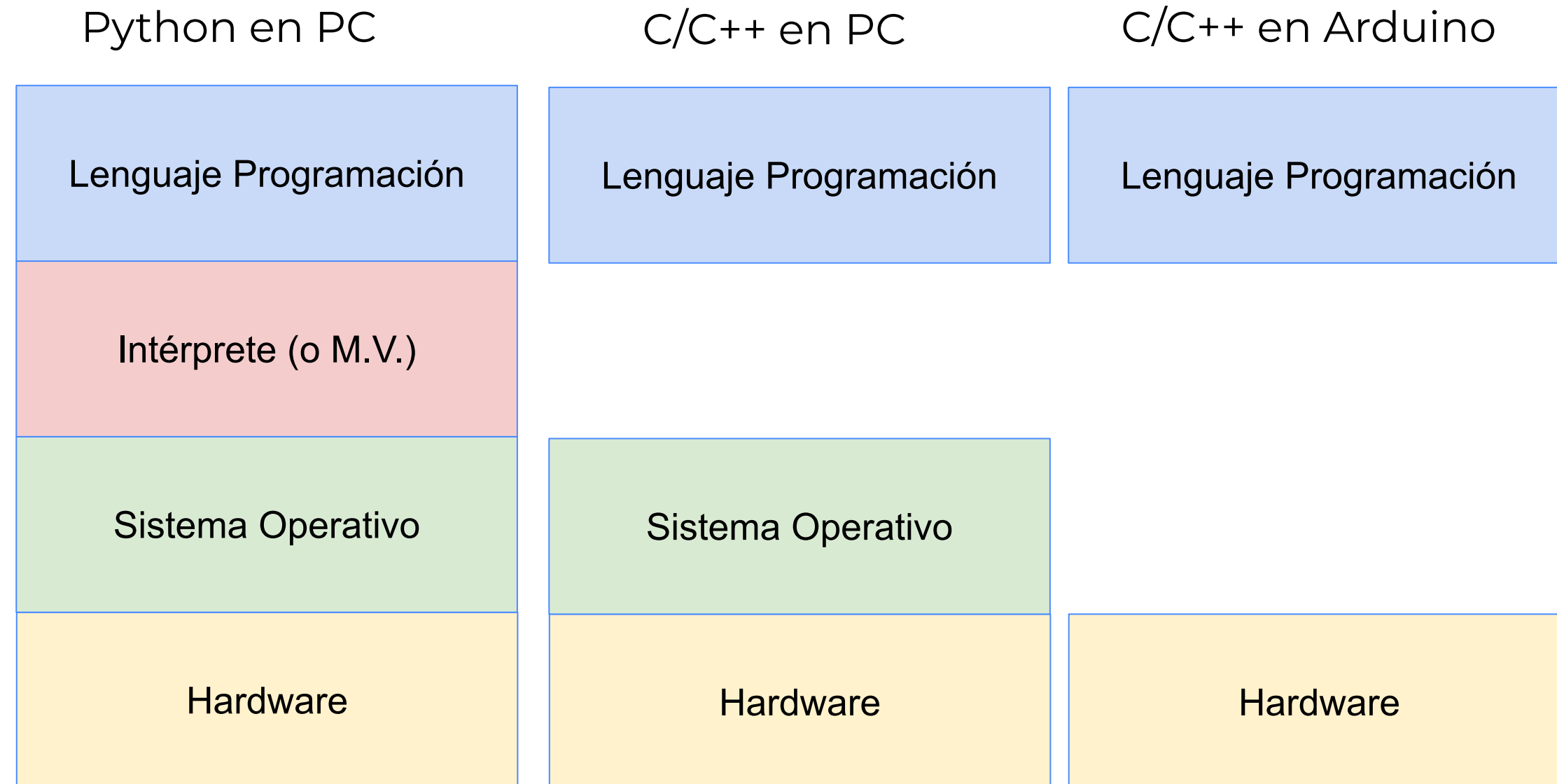
Implementación práctica

01 **Arquitectura de un arduino**

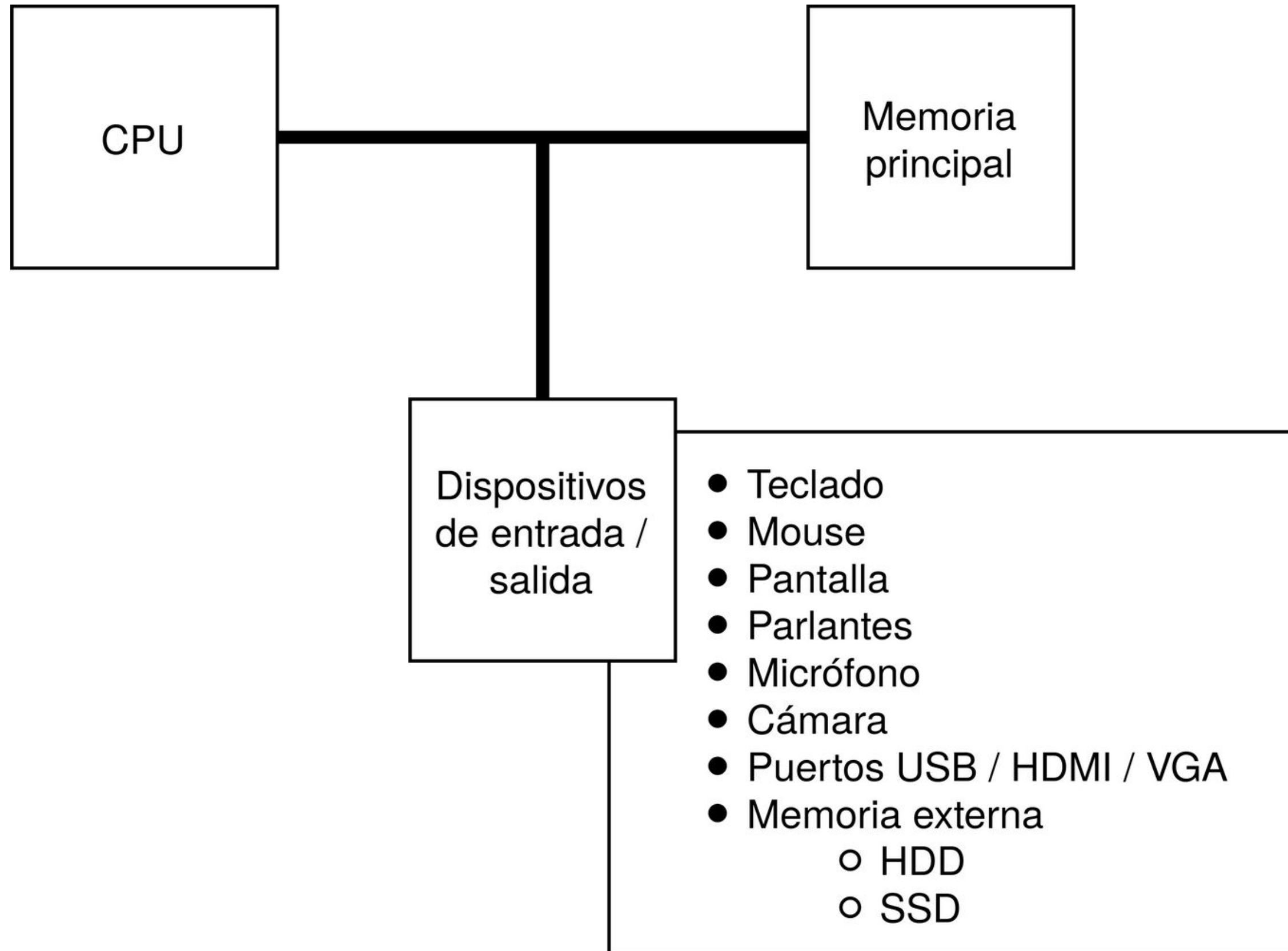


Sistemas embebidos

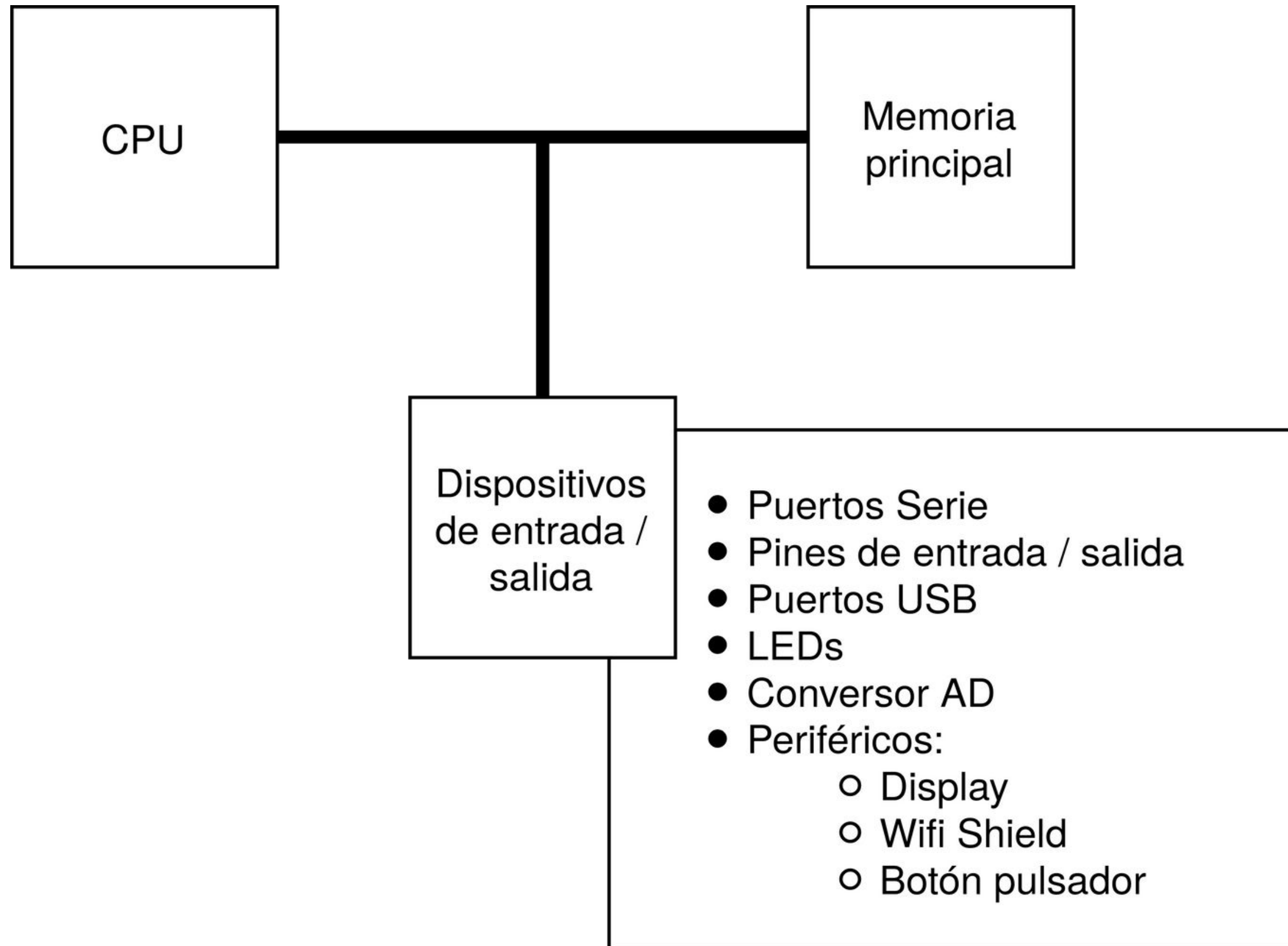
- Definición
- Características
 - Arquitectura reducida
 - Bajo costo, bajo consumo
 - Poco poder de cálculo
 - Función específica
- Ejemplos:
 - Microbit
 - **Arduino**
 - Raspberry PI
 - ESPRESSIF (ESP32)
- Aplicaciones
 - Dispositivos médicos portátiles
 - Internet de las cosas (IoT)
 - Cámara de seguridad



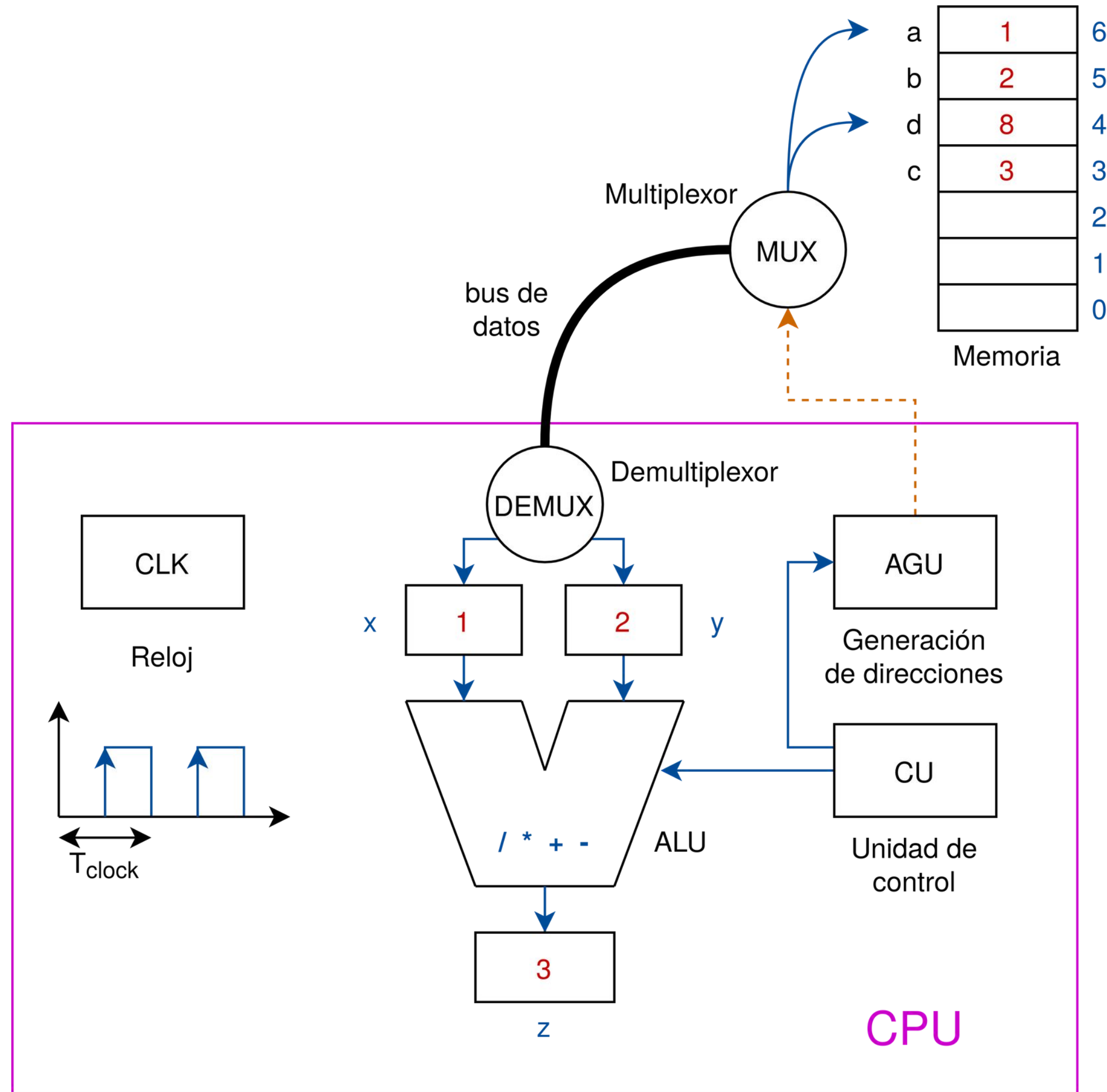
Arquitectura de una PC



Arquitectura de un arduino



CPU



Resumen



- **Central:**
 - Microcontrolador ATMEL
 - Clock
 - Memória **limitada**
- **I/O:**
 - Pines de entrada / salida
 - Conversor AD
 - Puerto USB
 - LEDs
- **Software:**
 - IDE
 - Programación en C/C++

02

Características de cada placa



Tipos

Arduino UNO:

Se basa en un microcontrolador Atmel ATmega320 de 8 bits a 16Mhz que funciona a 5V.

32KB son correspondientes a la memoria flash, 2KB de SRAM y 1KB de EEPROM. Las salidas pueden trabajar a voltajes superiores, de entre 6 y 20V pero se recomienda una tensión de trabajo de entre 7 y 12V.

Arduino Mega:

El microcontrolador que lo maneja es un ATmega2560. Este chip trabaja a 16Mhz y con un voltaje de 5v. Sus capacidades son superiores al ATmega320 del Arduino UNO, aunque no tan superiores como las soluciones basadas en ARM. Este microcontrolador de 8 bits trabaja conjuntamente con una SRAM de 8KB, 4KB de EEPROM y 256KB de flash (8KB para el bootloader).

Las facultades de esta placa se asemejan al Due, pero basadas en arquitectura AVR en vez de ARM.

Arduino Due:

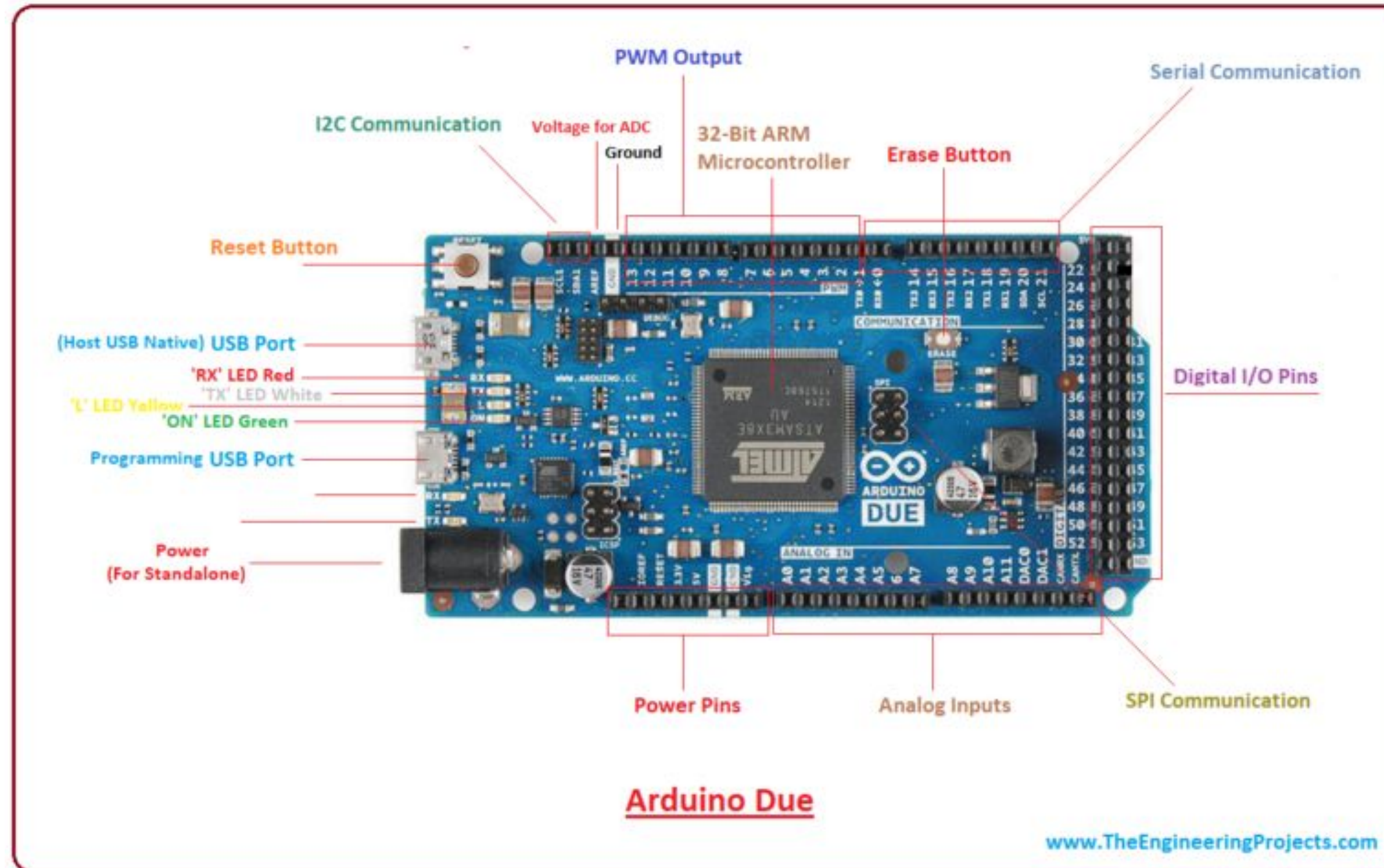
Es una placa con un microcontrolador Atmell SAM3X8E ARM Cortex-M3 de 32 bits. Este chip trabaja a 84Mhz (3,3v) y aporta una potencia de cálculo bastante superior a los anteriores. Idóneo para todos proyectos que requieran una alta capacidad de procesamiento.

La memoria SRAM es de 96KB, para el almacenamiento se dispone de 512KB de flash. En cuanto a soporte de voltajes en intensidades es idéntica a UNO.

Tipos

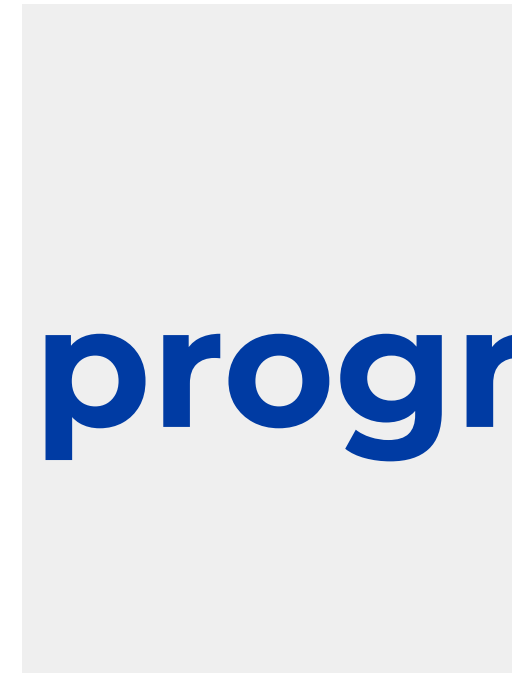
	Arduino UNO	Arduino Mega 2560	Arduino Due
Largo de palabra	8 bits	8 bits	32 bits
Microcontrolador	ATmega320	ATmega2560	SAM3X8E Cortex-M3
Frecuencia de Clock	16 MHz	16 MHz	84 MHz
Memoria flash	32 Kb	256 Kb	512 Kb
Memoria SRAM	2 Kb	8 Kb	96 Kb

Arduino DUE

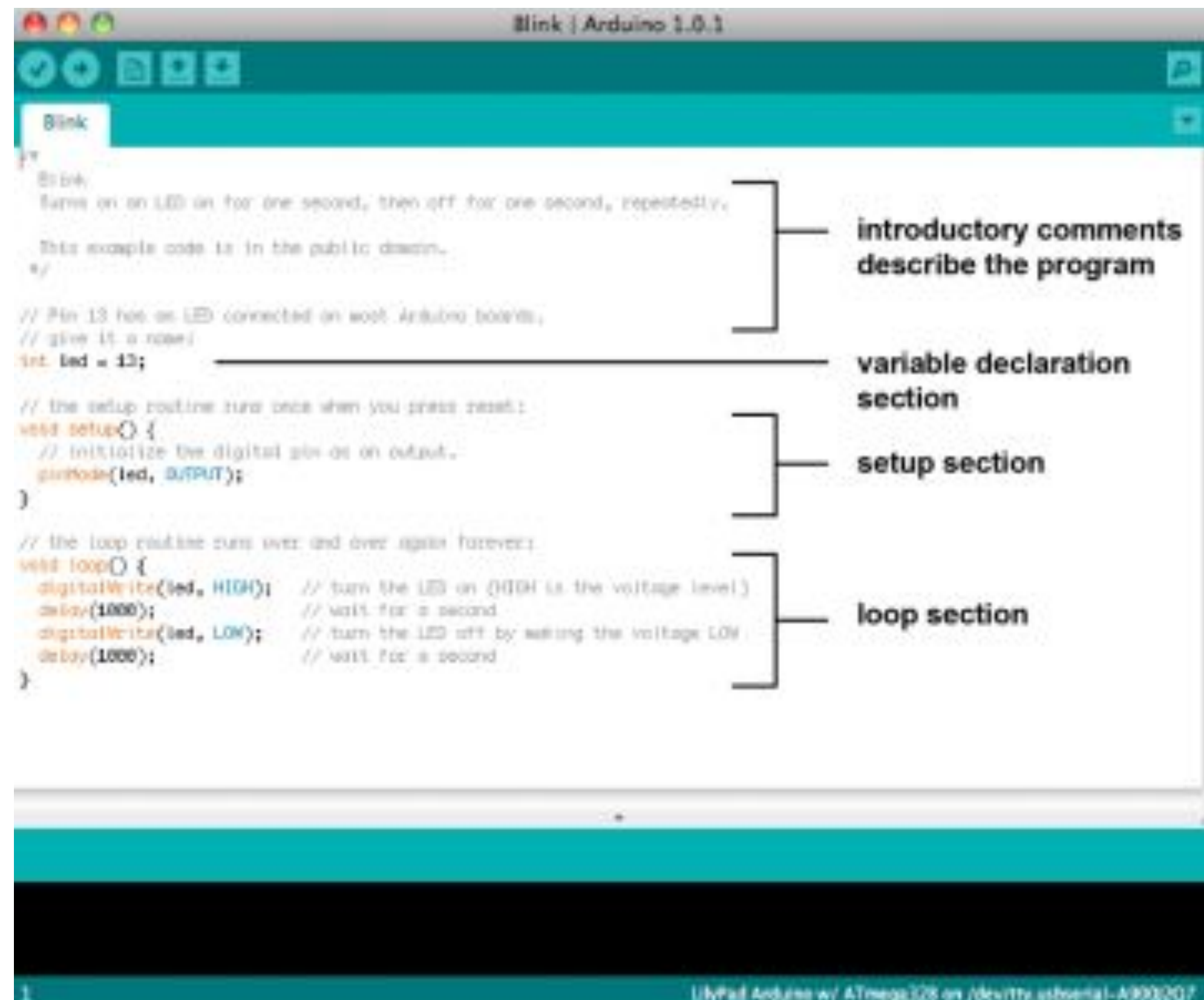


03

Análisis de un programa



Partes del sketch



The image shows a screenshot of the Arduino IDE interface with the 'Blink' sketch open. The code is annotated with brackets on the right side, identifying its different sections:

- introductory comments describe the program**: This section includes the multi-line comment at the top of the sketch that describes the program's purpose and notes that the code is in the public domain.
- variable declaration section**: This section is the line of code that declares the variable `int led = 13;`.
- setup section**: This section is the `void setup() { ... }` function, which initializes the digital pin as an output.
- loop section**: This section is the `void loop() { ... }` function, which contains the code that repeatedly turns the LED on and off with a one-second delay.

```
/*  
 * 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  
}
```

Uso de periféricos I/O

- Puerto digital
 - Ejercicio 1 -> Encendido de un LED
- Puerto Serie
 - Ejercicio 1 -> Funcionamiento del puerto serie
 - Ejercicio 2 -> ¿Qué ocurre si no se inicializa la conexión?
 - Ejercicio 3 -> ¿Qué ocurre si se cierra la conexión?
- Frecuencia de Muestreo
 - Ejercicio 1 -> ¿Cómo se fija la frecuencia del loop?
 - Ejercicio 2 -> ¿Qué ocurre al incrementarla?

Uso de periféricos I/O

- Puerto analógico
 - Lectura de una señal
 - Generación de una señal

04 Implementación práctica



◆ **Ejercicio 1** (Implementación de un control de volumen)

- (a) Realice un programa que tome muestras de la señal de entrada y las amplifique por un factor constante.
- (b) Defina las señales de entrada a trabajar.
- (c) Calcule la frecuencia de muestreo necesaria para representar correctamente las señales.
- (d) Verifique el correcto funcionamiento del programa.

Gracias

¿Preguntas?

Renato Sosa Machado



renato.sosast@gmail.com

Lucía Lemes



llemes@cup.edu.uy

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**