



INTRODUCCIÓN A MICROCONTROLADORES

- Introducción - Qué es un microcontrolador
- Arquitecturas/Familias actuales en sistemas embebidos
- Circuitos Integrados
- Concepto de *Dev board*
- Conversor Analógico/Digital (ADC)
- Protocolos de Comunicación Serie (*I2C, SPI, UART*)
- Programación en 
- Frameworks de Software en sistemas embebidos
- Programación en *Arduino* 
- Proyectos: Monitor Serie, Lectura BME280, GPS, TOUCH, ...

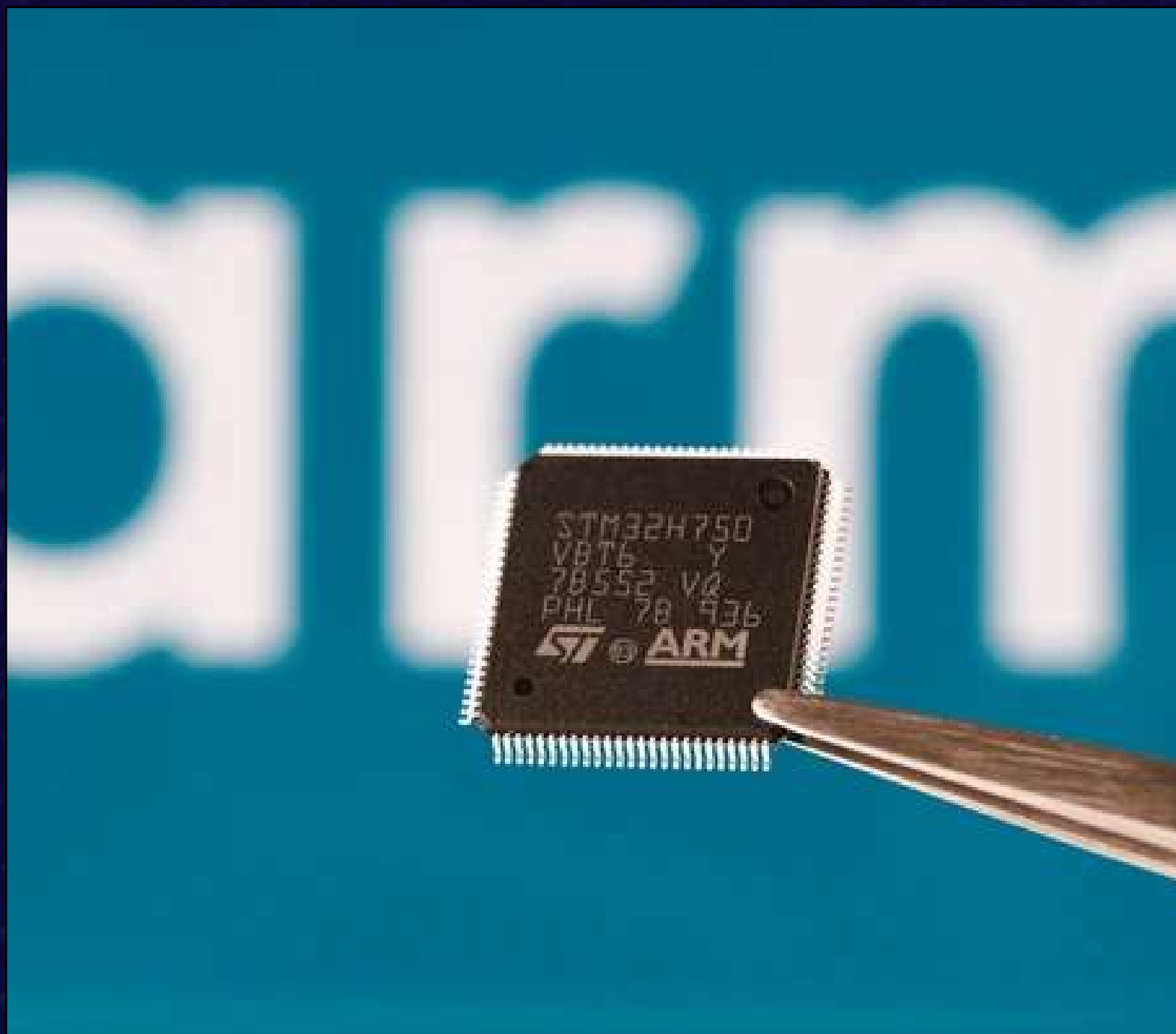
QUE ES UN MICROCONTROLADOR?

Un **microcontrolador** es un tipo de **circuito integrado** que puede programarse.

Podéis pensar en ello como un "*ordenador*" con recursos muy limitados.

Sus recursos son tan limitados que **no se consideran un ordenador** (no puedes cargarle un **sistema operativo** (OS))

Existen ordenadores pequeños contenidos en una sola placa, denominados **Single Board Computers** (SBC). (Ejemplo más famoso las **Raspberry Pi**)

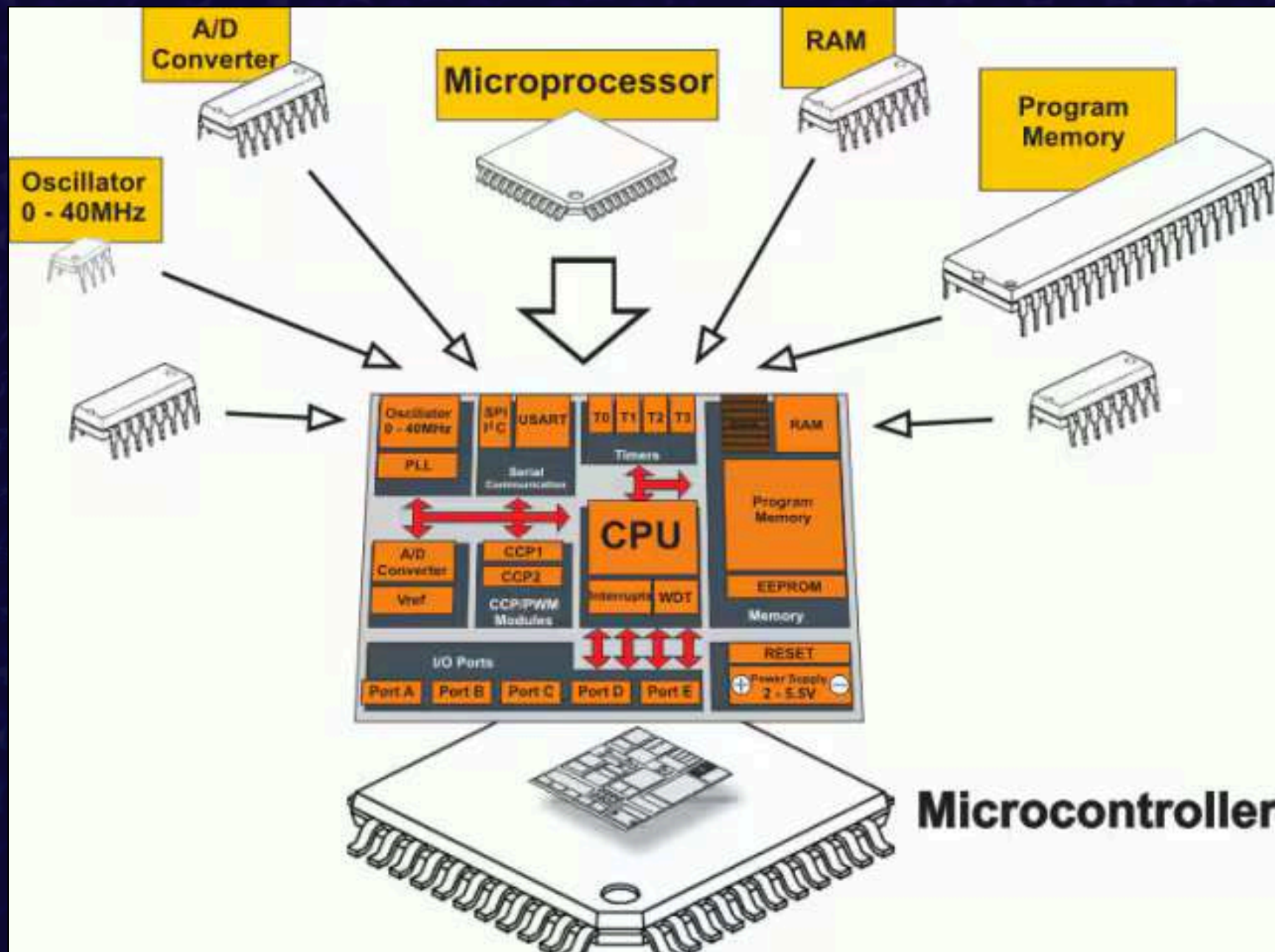


CHIP MICROCONTROLADOR

QUE ES UN MICROCONTROLADOR?

Los **microcontroladores** normalmente cuentan con cierto tipo de **componentes**, todos **integrados en un chip**:

- **Microprocesador**: La CPU en un solo integrado, ejecuta el código en la memoria de programa.
- **Memoria de Programa**: Memoria interna para almacenar código.
- **Pines de entrada y salida de propósito general (GPIO)**:
- **Oscilador**: Para generar señales de reloj.
- **Relojes**: Para contar el paso del tiempo de forma precisa (tareas temporizadas).
- **RAM**: Memoria volátil, podéis pensar en ella como “memoria a corto plazo”. No se queda guardada, la usas en el momento.
- **ADC**: Conversor Analógico digital para convertir señales analógicas en digitales.



COMPONENTES DE UN MICROCONTROLADOR

arm

AVR[®]

ESPRESSIF

RISC-V

ARQUITECTURAS ACTUALES En sistemas embebidos



ARM es una arquitectura **RISC** de **32 bits**. Es muy popular y se está usando mucho en ordenadores portátiles y tablets. Ejemplo de microcontrolador: Familia **STM32**



AVR es la familia de microcontroladores de **8 bit** que usan los modelos de **Arduino** (UNO, Mega, Nano...).



Espressif es una empresa china que desarrolla la familia de microcontroladores de **32 bits ESP32**. El **ESP32** original usa la arquitectura **xtensa**, pero actualmente hay otros modelos que usan **RISC-V**. Destacan por contar con Wi-Fi/BLE integrado.



RISC V es un juego de instrucciones Open Source que está teniendo mucha adopción recientemente para competir con ARM.

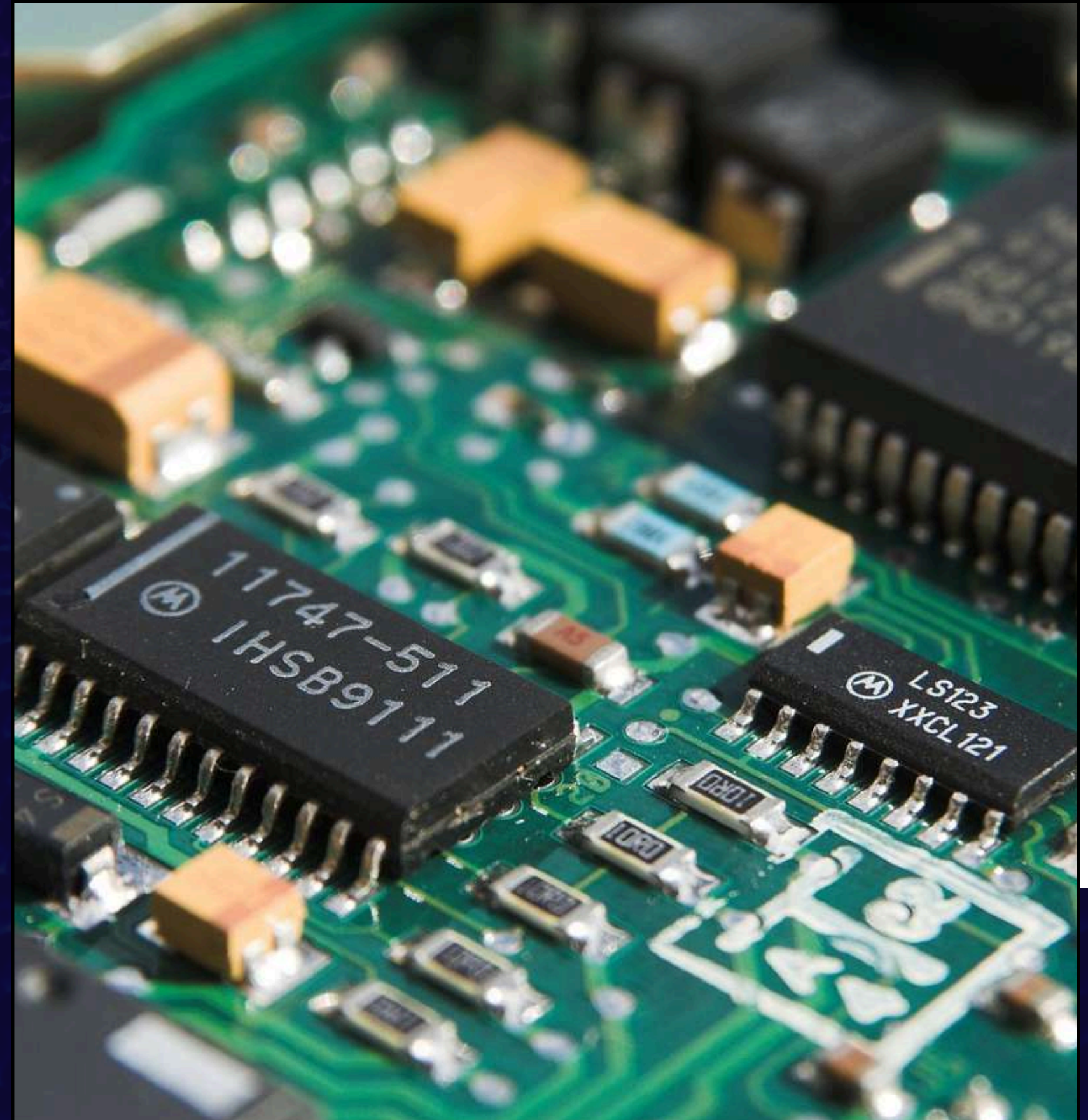
CIRCUITOS INTEGRADOS

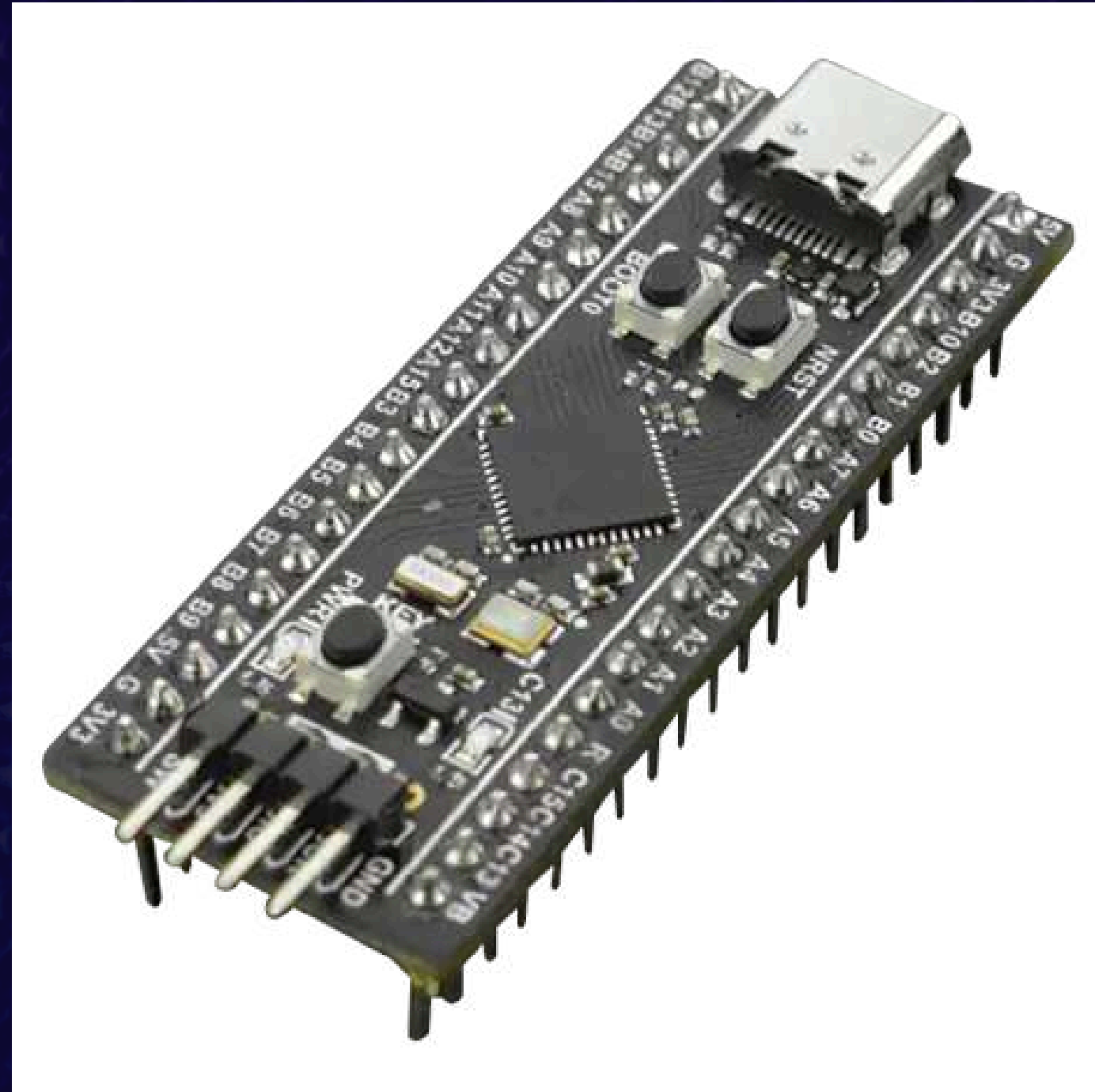
En la actualidad es raro realizar circuitos electrónicos a mano. Salvo **componentes pasivos** (resistencias, condensadores, diodos...) los circuitos electrónicos se realizan combinando **circuitos integrados** que cumplen cometidos específicos.

En la imagen podemos ver varios circuitos integrados:

- **11747-511**: En desuso, no encontré referencias.
- **LS123**: Multivibrador monoestable. (En desuso)

Además se pueden ver varias **resistencias**, **condensadores** y **bobinas** de **SMD**.





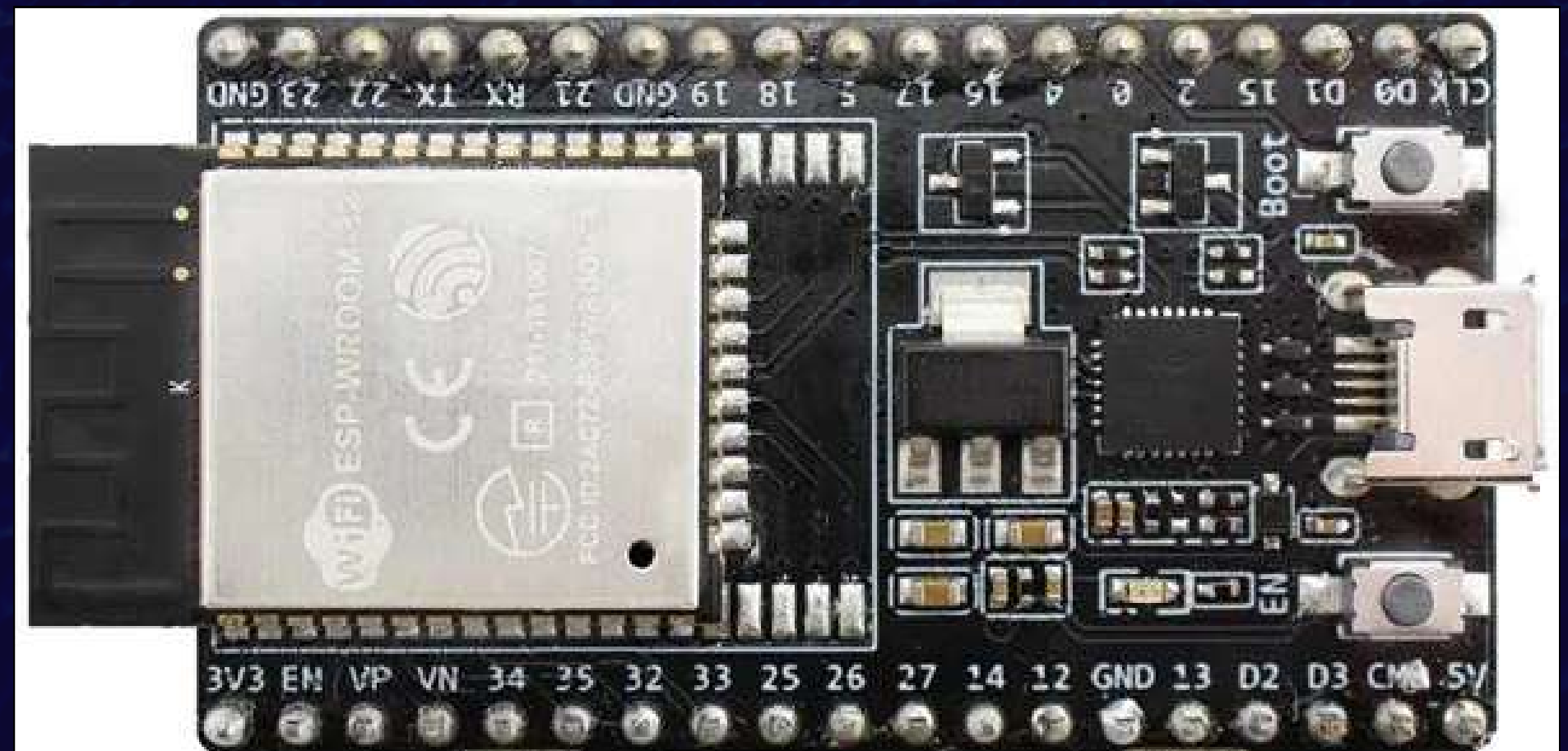
STM32F411 "BLACK PILL"

DEV BOARD

Una **dev board** no es más que una placa de circuito impreso (**PCB**) con todos los **componentes necesarios** para el funcionamiento correcto de un **microcontrolador**.

(Regulador para alimentación, diodos, resistencias, condensadores de desacoplo, pulsadores de reset, acceso a pines...)

ESP32 DEVKIT C



ADC

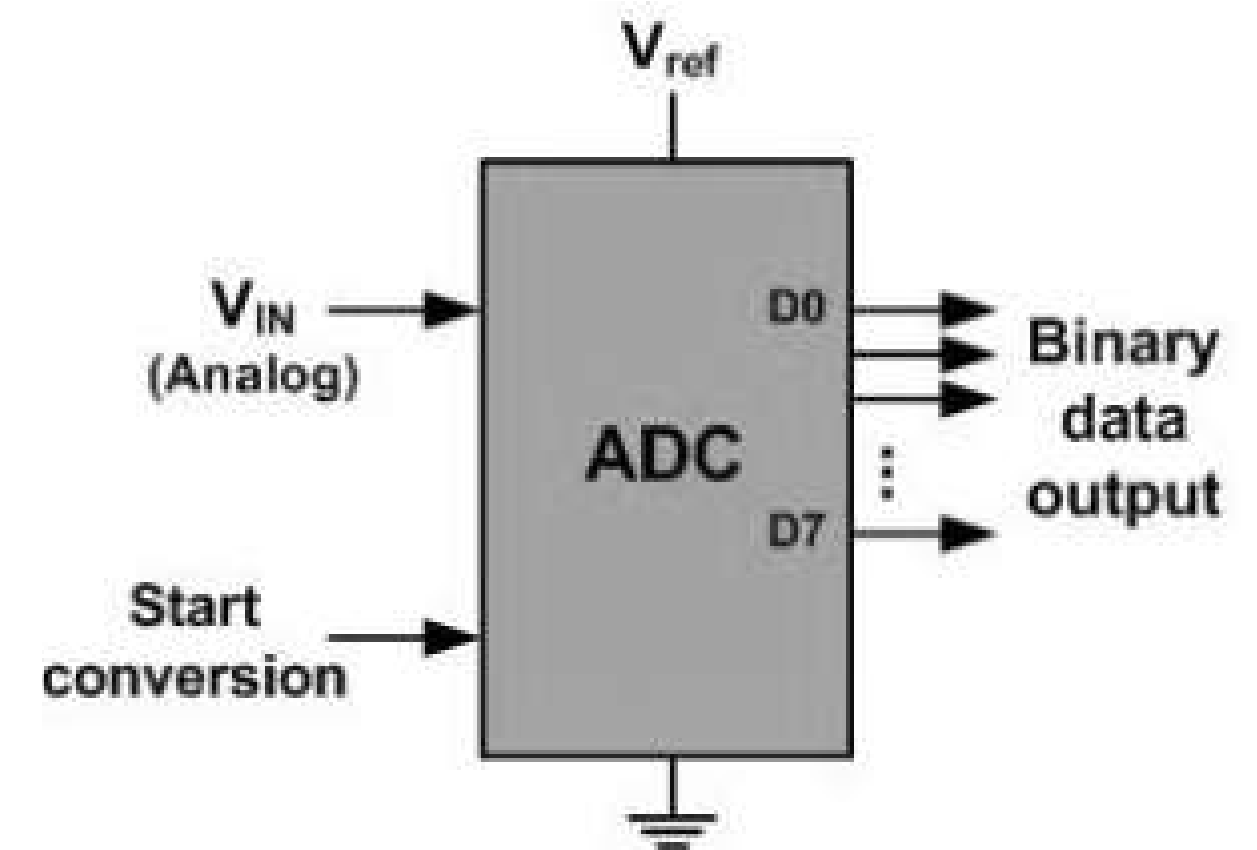
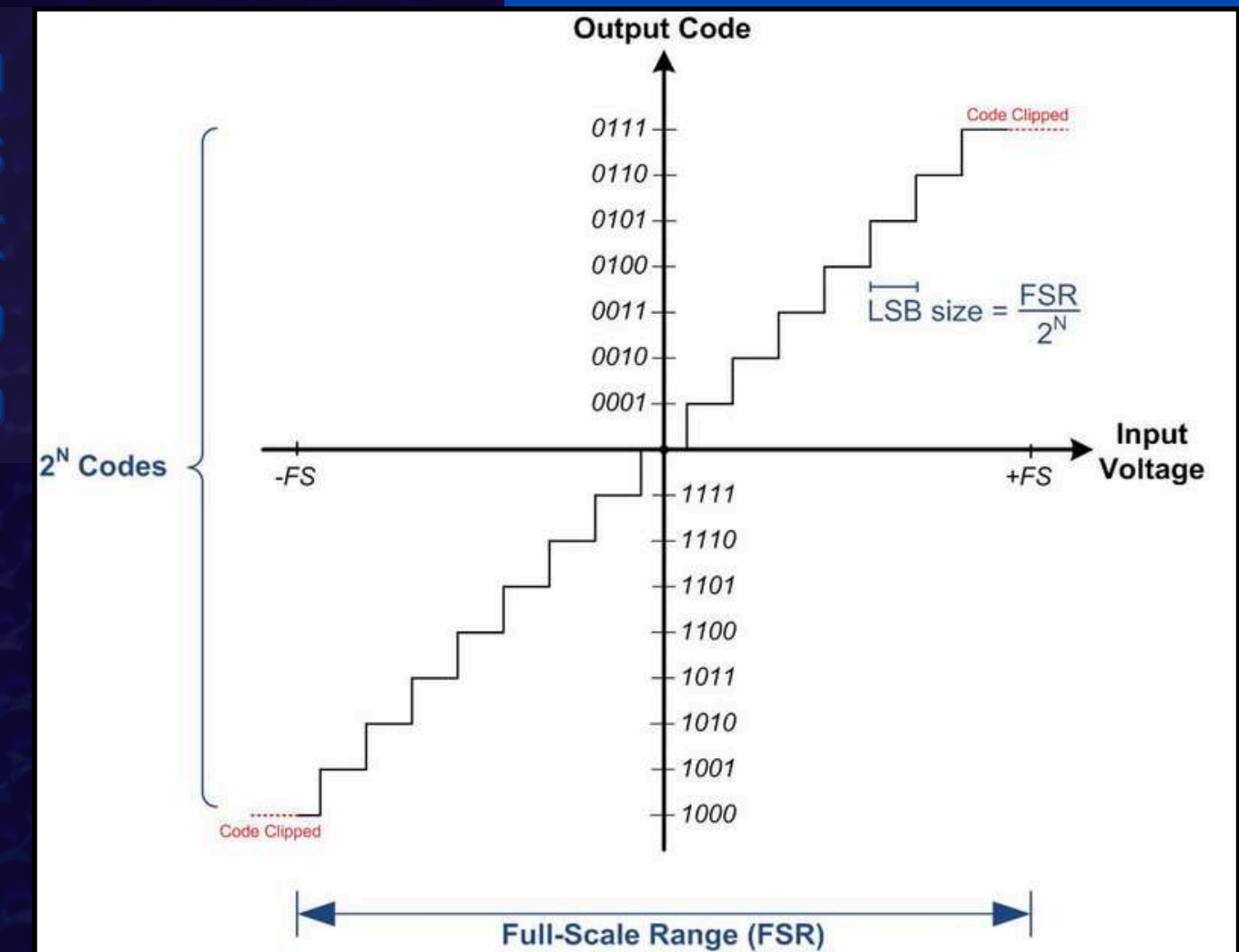
GRAFICA
CODIGOS VS
ESCALA DE
RANGO
COMPLETO

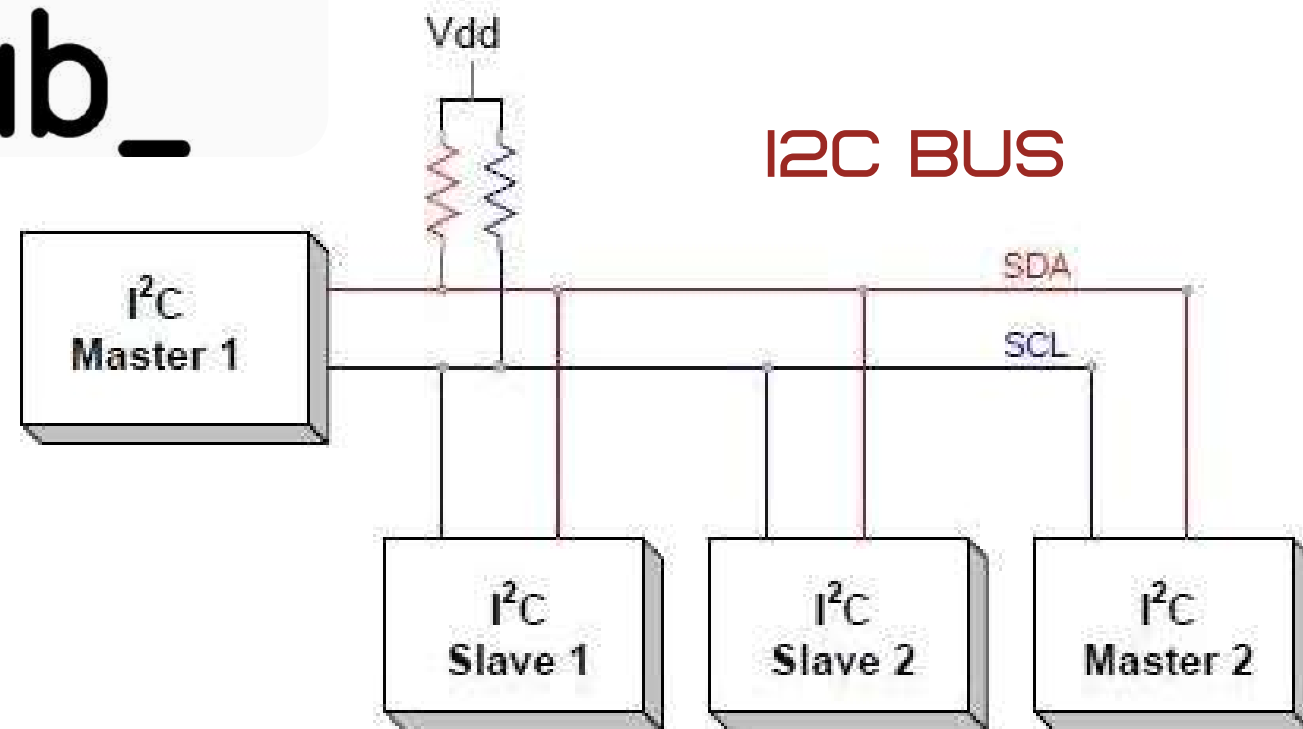
Para convertir las señales analógicas (variable continua) en señales digitales (variable discreta) se utilizan unos componentes llamados **Conversores Analógico-Digitales (ADC)**.

Los parámetros más importantes de un **ADC** son:

- **nº de bits (n)**: el número de bits de un convertidor te delimita el número de códigos posibles para representar el rango de una señal. 2^n códigos
- **rango** de trabajo: Diferencia entre la **tensión** máxima y mínima del convertidor.

Es importante destacar que los **ADC** siempre usan una **referencia de voltaje**, ya sea interna o externa.

CONEXIONADO
ADC



PROTOSCOLOS DE COMUNICACION SERIE

I²C

Comunicación **serie síncrona** de **2 pines**, **half dúplex** (comunicación en ambos sentidos pero no a la vez).

- 2 cables, 1 bus.
- Necesita resistencias de polarización (pull-up)
- 127 direcciones posibles en un bus
- Distancias cortas, sensible a interferencias.

SPI

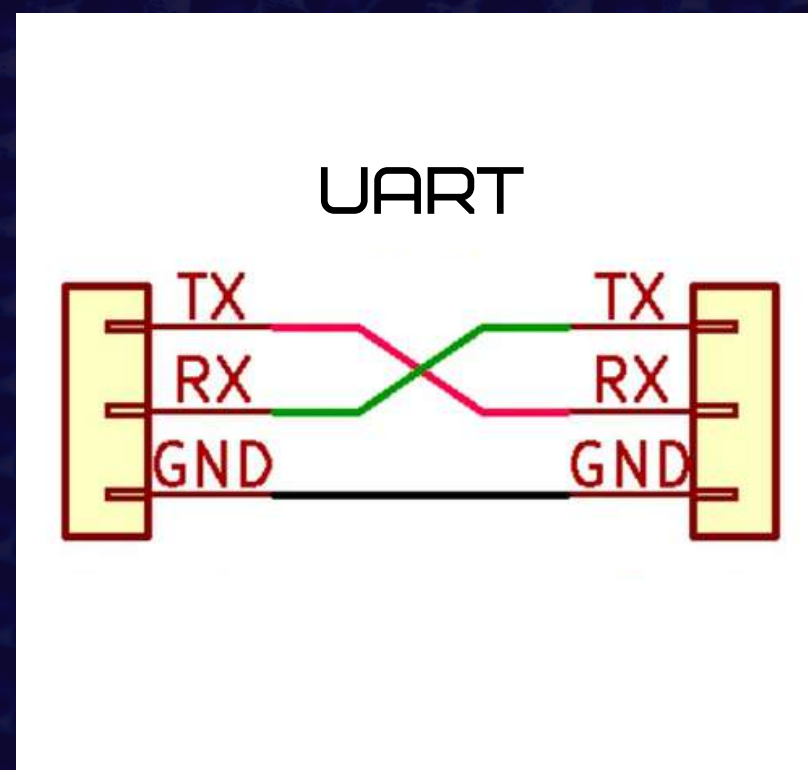
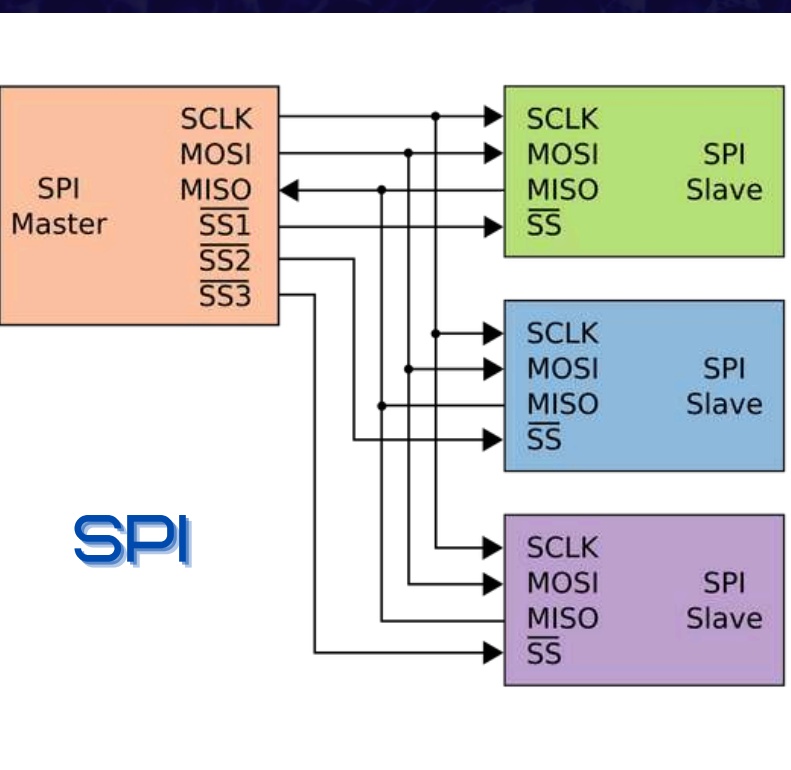
Comunicación **serie síncrona** de **4 pines**, **full dúplex** (comunicación en ambos sentidos simultáneamente)

- 4 cables, 1 bus
- No necesita resistencias de polarización
- Es muy rápido, se suele usar en tarjetas SD.

UART

Comunicación serie asíncrona de 3 pines, full dúplex.

- 3 cables, 2 dispositivos
- No envía señal de reloj → Configurar baudios
- Se puede configurar como TX, RX, semidúplex o full dúplex.
- Más resistente a interferencias.



media lab_ PROGRAMACION EN C

C es un lenguaje de programación **compilado** de **tipado estático** (*static typing*). Suele utilizarse ampliamente como **lenguaje de sistemas**. (*Sistemas operativos, microcontroladores, ...*)

- En C las instrucciones se empiezan a leer de arriba abajo.
- Cada instrucción acaba con un **;**
- El programa empieza con la función **main()** (**main()** es el *punto de entrada*)
- En microcontroladores suele usarse el **bucle de ejecución continua**. (Imagen superior)

HELLO
WORLD EN C

```
1 // Programa bucle de ejecución continua
2 #include <stdio.h>
3 #include <stdbool.h>
4
5 bool SENSOR_CONFIGURADO = false;
6
7 void configurar_sensor();
8 float dato = 0;
9
10 int main(){
11     // setup
12     configurar_sensor();
13     SENSOR_CONFIGURADO = true;
14     // bucle de ejecución continua
15     while(1){
16         dato = leer_sensor();
17         printf("El dato es: %f", dato);
18     }
19     return 0;
20 }
```

BUCLE DE EJECUCION
CONTINUA

```
// Programa hello world en C
#include <stdio.h>

int main(){
    printf("Hello world");
    return 0;
}
```


FRAMEWORKS



Arduino: es un **framework** de software que facilita escribir **software** para **microcontroladores**. En principio enfocado para *hobbyistas* cada vez tiene más uso profesional.

- **Ventajas:** Facilidad de implementación, amplio soporte de la comunidad, "*multiplataforma*".
- **Desventajas:** Te esconde detalles, poca adaptabilidad.



ESP-IDF: Framework oficial de la familia de microcontroladores **ESP32**. Más avanzado.

- **Ventajas:** soporte del fabricante, acceso nativo a librerías, mayor control de periféricos, coprocesador ULP, ...
- **Desventajas:** más difícil, sólo funciona para **ESP's**.



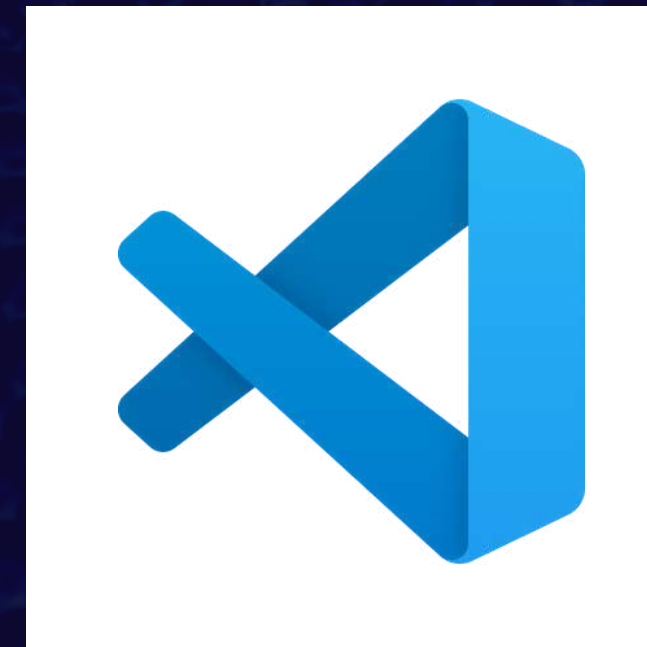
GNU C/C++ for Arm: Compilador de código libre para procesadores basados en **ARM**.

- **Ventajas:** Ecosistema maduro, sintaxis más similar a C "clásico",
- **Desventajas:** Compilación lenta, código no tan optimizado como compiladores de pago (**Keil**, etc...)

IDE'S
PARA CADA
FRAMEWORK



Arduino IDE
(Arduino Framework)



VS Code (ESP-IDF)



CubeIDE (GNU ARM)

media lab_ PROGRAMACION EN ARDUINO

Arduino es un framework de una versión no-estándar de **C++**. Inicialmente diseñado para la programación de *dev boards* diseñadas por la empresa del mismo nombre para *hobbyistas*, actualmente cuenta con soporte para un amplio número de **microcontroladores**.

Como mencionábamos anteriormente **Arduino** te oculta detalles y llama a **main()** internamente, dejando al usuario definir dos **funciones**:

- **setup()**: Función que se ejecuta una vez al inicio del programa.
- **loop()**: Función que se ejecuta en bucle constantemente.

Teniendo esto en cuenta estamos haciendo un **bucle de ejecución continua**.

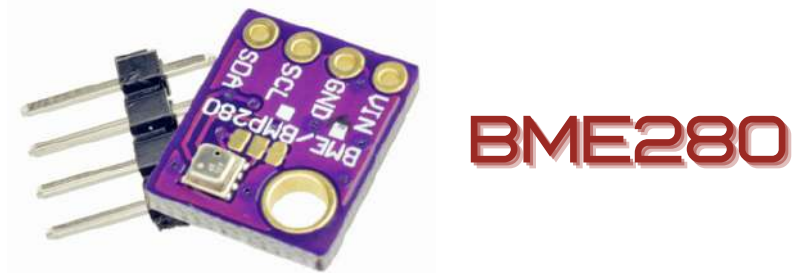
PROGRAMA
POR DEFECTO

```
10 #if defined(ESP8266)
11   #include <ESP8266WiFi.h>
12   #include <ESPAsyncTCP.h>
13 #elif defined(ESP32)
14   #include <WiFi.h>
15   #include <AsyncTCP.h> // https://github.com/ESP32Async/AsyncTCP
16 #endif
17 #include <ESPAsyncWebServer.h> // https://github.com/ESP32Async/ESPAsyncWebServer
18 #include <WebSerialLite.h> // https://github.com/asjdf/WebSerialLite
19
20
21 const int PORT = 80;
22 AsyncWebServer server(PORT);
23 // IP Estática
24 const IPAddress LOCAL_IP(192,168,4,1);
25 const IPAddress GATEWAY(192,168,1,1);
26 const IPAddress SUBNET(255,255,255,0);
27
28 const char* SSID = "Webserial"; // WiFi AP SSID
29 const char* PASSWORD = ""; // WiFi AP Password
30
31 /* Message callback of WebSerial */
32 void recvMsg(uint8_t *data, size_t len){
33   WebSerial.print(String("Received Data...") + "\n");
34   switch(data[0]){
35     case 's':
36       WebSerial.print(String("S COMMAND") + "\n");
37       break;
38     case 'd':
39       WebSerial.print(String("D COMMAND") + "\n");
40       break;
41     default:
42       WebSerial.print(String("DEFAULT COMMAND") + "\n");
43       break;
44   }
```

```
void setup() {
  // put your setup code here, to run once:
}

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

CODIGO COMPLICADO CON
MUCHAS COSAS



BME280



**GPS +
ANTENA**



**ESP32
DEVKIT C**

Mediante el uso de Arduino IDE: 

1. Programar la lectura de la **temperatura** y **presión** cada 5 segundos con un sensor **BME280** (I2C).
2. Configurar un pin **TOUCH** para que funcione como un “pulsador”. **Mostrar en pantalla** (*terminal serie*) cada vez que se pulsa.
3. Hacer que el **ESP32** imprima en el **terminal serie** las **redes Wi-Fi** disponibles. Intentar obtener la hora de internet (buscar **NTP**).
4. Conectar dos **ESP32** entre sí con **bluetooth**. Enviar primero **mensajes fijos** y luego cadenas de texto que escribáis en el **terminal serie**.
5. Sacar los datos del **módulo GPS** para que te diga **el timestamp** y la **posición** (coordenadas) e imprimirlos en pantalla. (*Tarda en pillar el satélite en un cold-start*)

Librerías:

- **BME280:** para el sensor de temperatura.

<https://docs.arduino.cc/libraries/bme280/>

Arduino ESP32: Para los pines de sensor de toque (TOUCH), bluetooth, etc. *Ya incluido cuando descargas el board manager de ESP32.*

- **TinyGPS:** Para el GPS. <https://docs.arduino.cc/libraries/tinygps/>

Componentes:

- **ESP32: Microcontrolador** con funcionalidad **Wi-Fi/BLE**.
- **BME280: Sensor** de **temperatura, humedad** y **presión**. Funciona por **I2C** y se alimenta a **3.3V**.
- **GY-NEO-6MV2:** Módulo **GPS**.
- **Cable USB-A a micro-USB:** Para **alimentar** + **UART**.

FIN