

Internet de las Cosas

Máster Universitario en Inteligencia Computacional e
Internet de las Cosas.

Universidad de Córdoba

Bloque I: Visión técnica de IoT

Objetivos

- Conocer los protocolos y estándares más habituales, para las diferentes capas del modelo OSI, para desplegar soluciones IoT.
- Disponer de criterios efectivos para proponer soluciones válidas en cada caso.
- Adquirir una visión completa de lo que supone un despliegue IoT a nivel técnico.

Índice

10. Tecnología de las cosas

- a. UART
- b. SPI
- c. I2C
- d. OneWire
- e. ADC
- f. PWM

11. Protocolos a nivel Físico/Enlace

- a. El medio radioeléctrico
- b. 802.15.4
- c. Lora
- d. BLE
- e. UWB

12. Protocolos a nivel Transporte/Aplicación

- a. MQTT
- b. COAP

13. Bibliografía

Índice

10. Tecnología de las cosas

- a. **UART**
- b. **SPI**
- c. **I2C**
- d. **OneWire**
- e. **ADC**
- f. **PWM**

11. Protocolos a nivel Físico/Enlace

- a. El medio radioeléctrico
- b. 802.15.4
- c. Lora
- d. BLE
- e. UWB

12. Protocolos a nivel Transporte/Aplicación

- a. **MQTT**
- b. **COAP**
- c. **ZigBee**

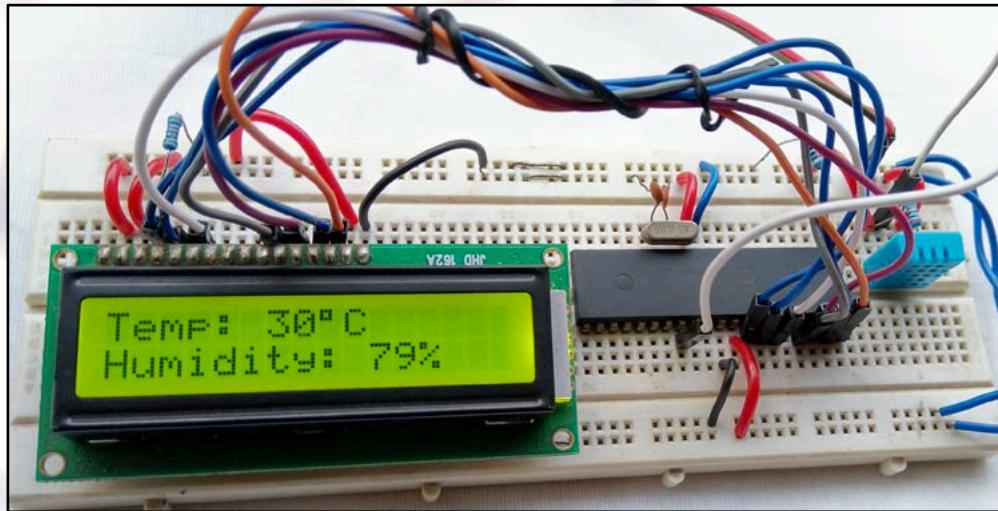
13. Bibliografía

Tecnología de las cosas

¿Qué son las cosas?

Las "cosas" son **sistemas empotrados**

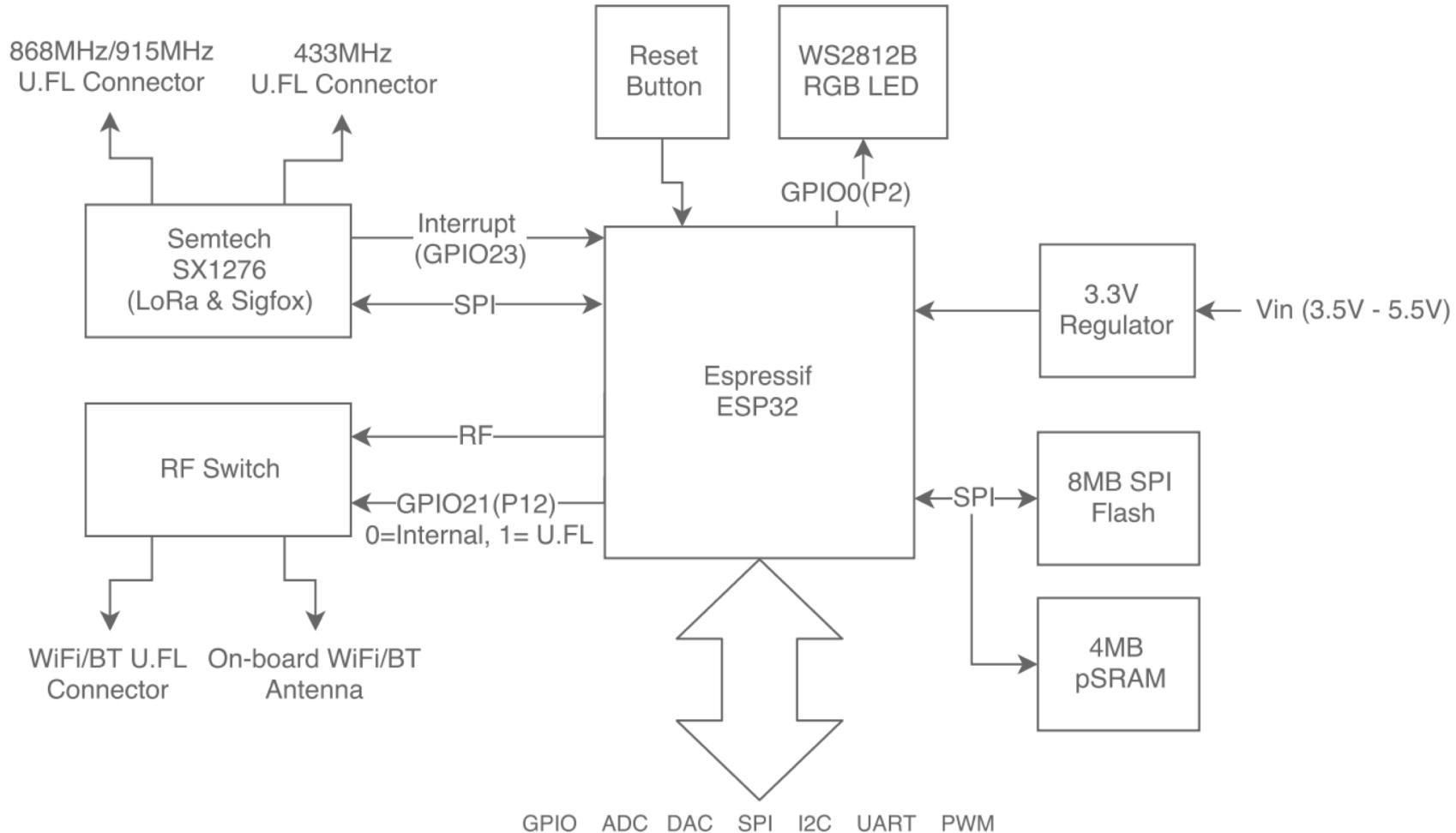
1. Tienen un propósito específico.
2. Se diseñan con los recursos necesarios.
3. Interfaz mínima y a medida.
4. Deben ofrecer alta disponibilidad y confiabilidad.
5. A menudo satisfacen requisitos de tiempo real.
6. Suelen requerir integración de periféricos/componentes
7. A veces, requieren tener tamaño limitado



<https://circuitdigest.com/microcontroller-projects/interfacing-dht11-sensor-with-pic16f877a-microcontroller>

Tecnología de las cosas

Plataformas de sistemas empotrados (boards)



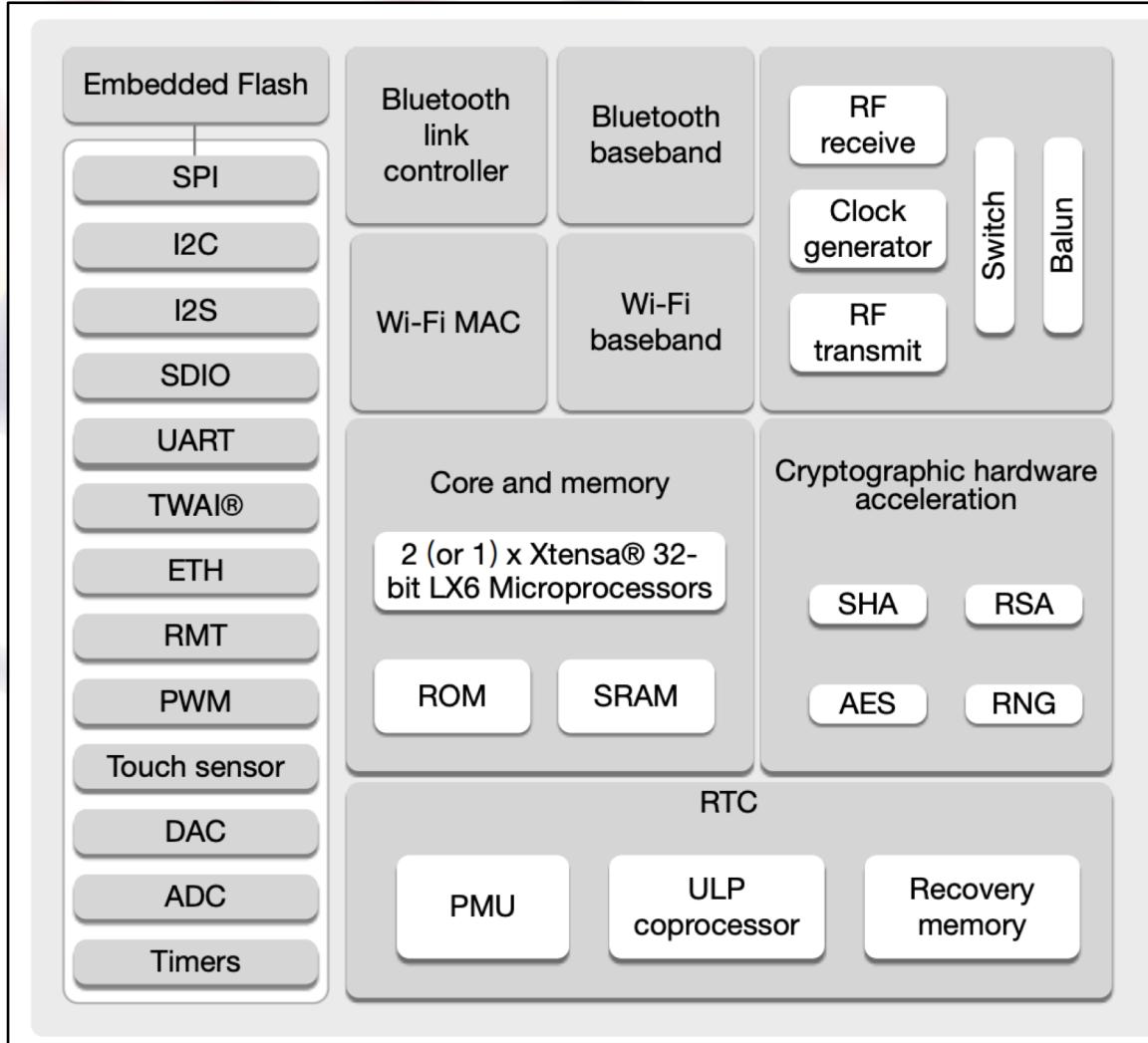
LoPy4

- MCU: ESP32 (SoC + MCU)
- Alimentación
- Flash y RAM externas
- Comunicación directa con MCU a través de diferentes puertos
- Driver SX1276 (Lora / Sigfox) vía SPI
- Conectores de antena para SX1276
- Botón de reset
- Led de actividad

https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf

Tecnología de las cosas

Sistemas on Chip (SoCs)



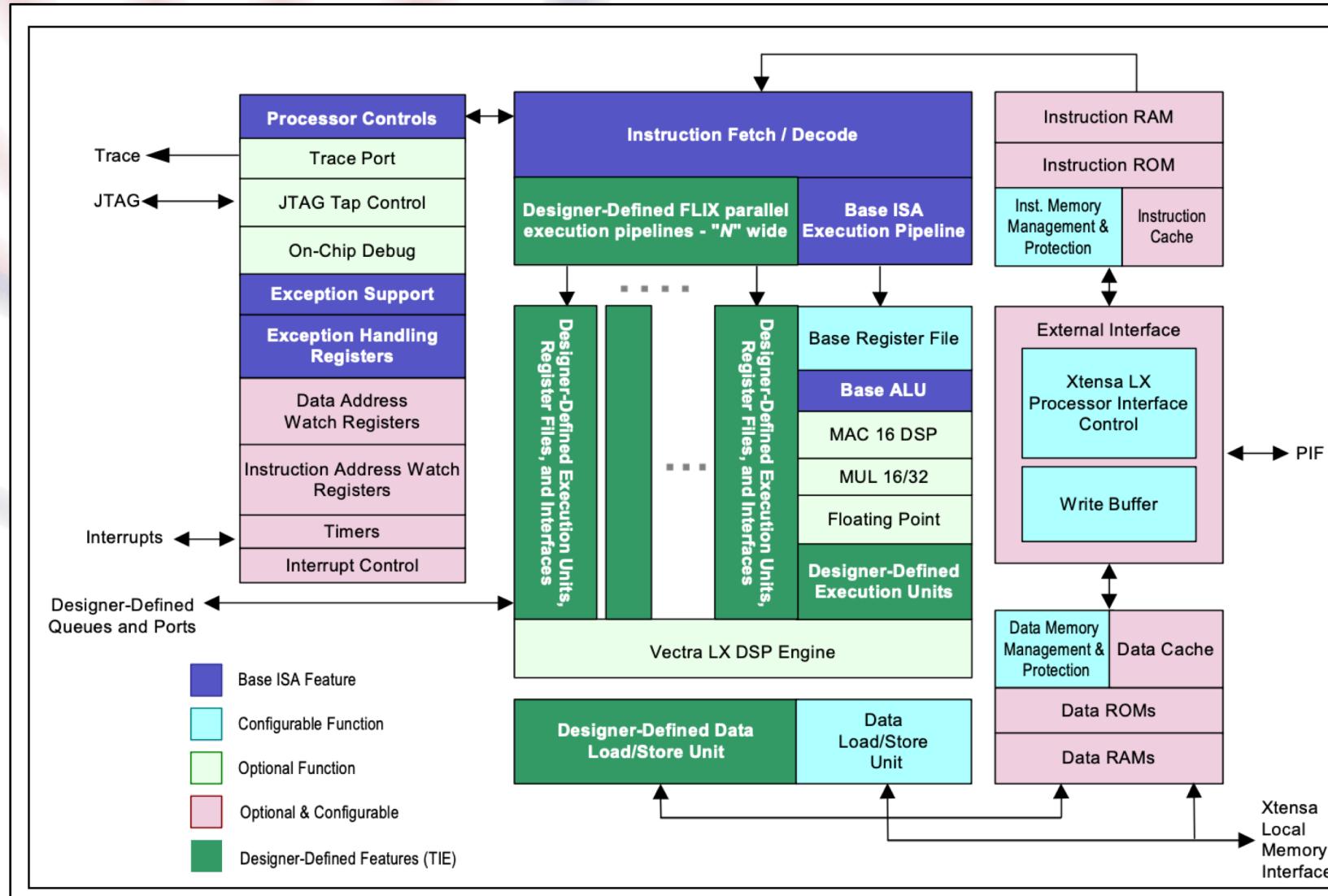
ESP32

- MCU: Xtensa®32-bit LX6
- ROM y SRAM integrada
- Memoria Flash integrada
- Controlador Wi-Fi
- Controlador Bluetooth
- Hardware para criptografía eficiente
- Reloj de tiempo real (RTC) con módulos de control de energía.
- Puertos para periféricos (SPI, I2C, UART, PWM, ADC, etc.)
- Timers hardware

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Tecnología de las cosas

Microcontroladores (MCUs)



XTENSA LX6

- Pipelines (CPU)
- RAM / ROM / Cache
- Interfaces de prog.
- Fuentes de interrupción
- Buffers de E/S
- Registros
- Buses del sistema
- Periféricos integrados
 - Timers
 - Canales DMA
 - Watchdog
 - UARTs
 - ADCs
 - Etc.

Tecnología de las cosas

Microcontroladores (MCUs)

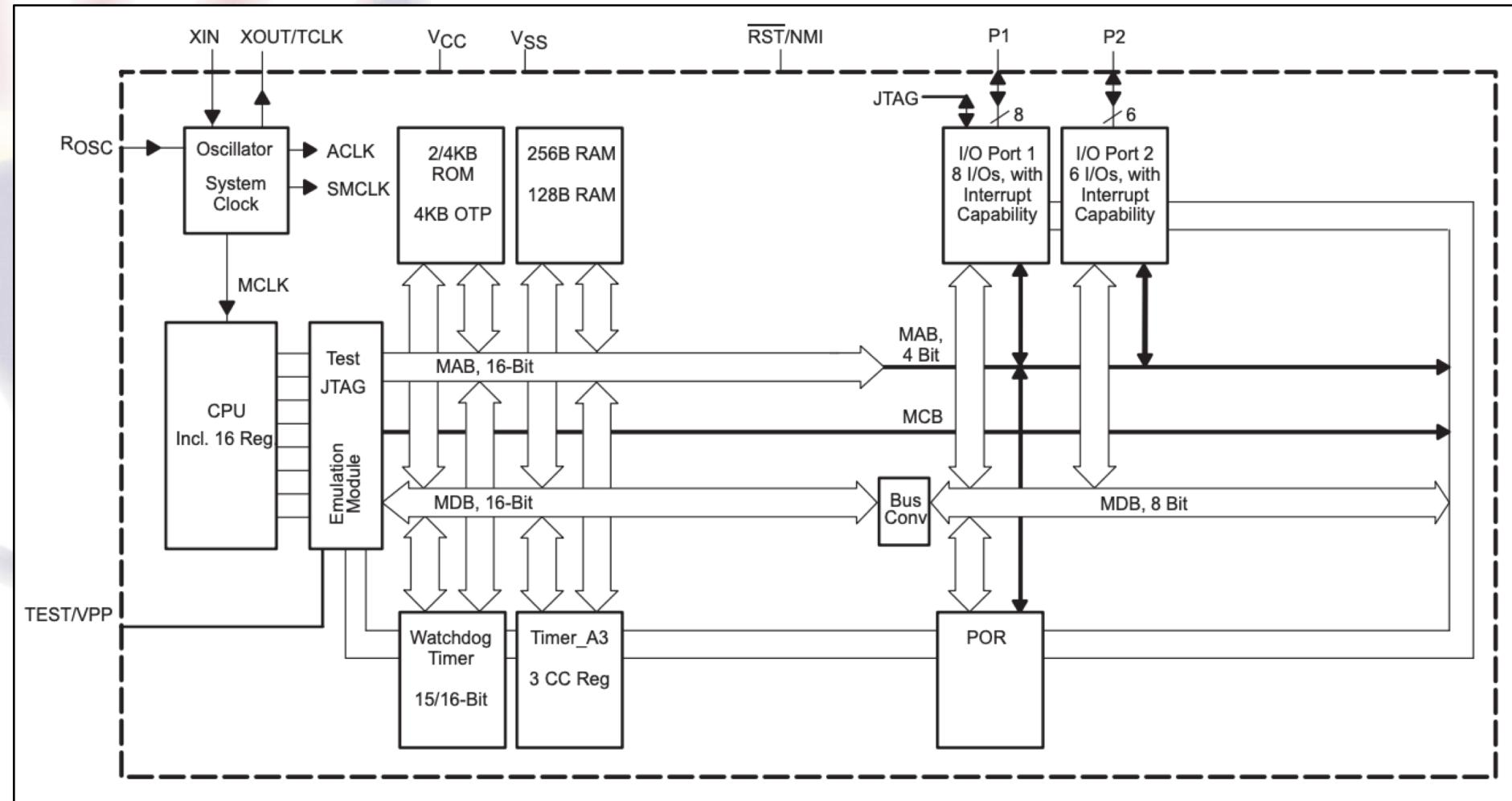
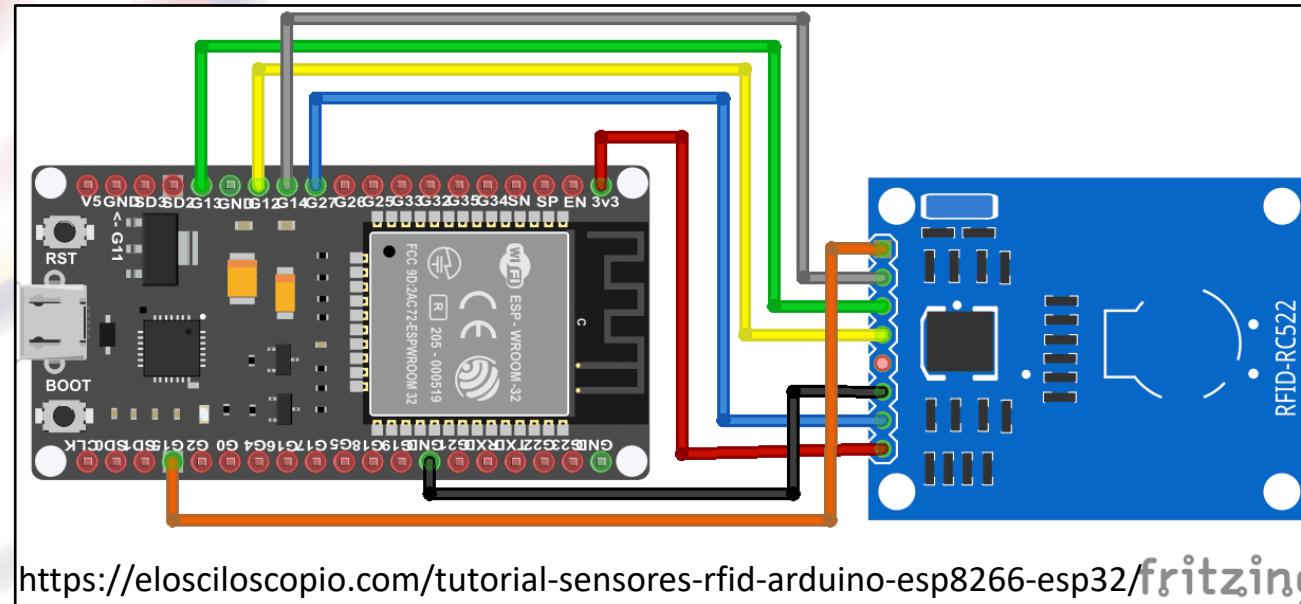


Diagrama funcional del MSP430C11x (TI)

Tecnología de las cosas

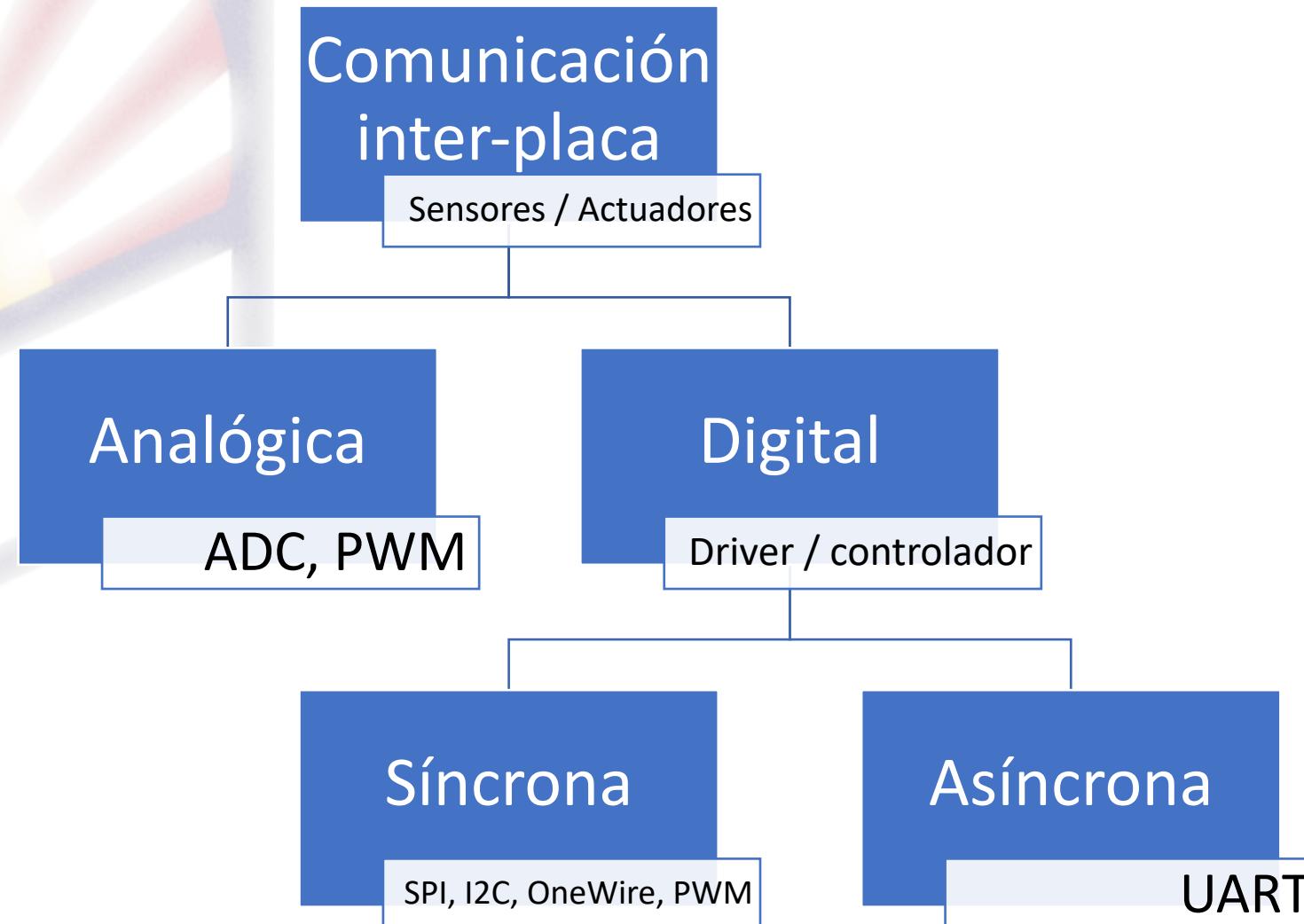
Comunicaciones inter-placa (periféricos)



¿Cómo integrar periféricos en nuestro sistema?
¿Cómo se programa la interfaz (leer / escribir)?
¿Cómo se programa esto?

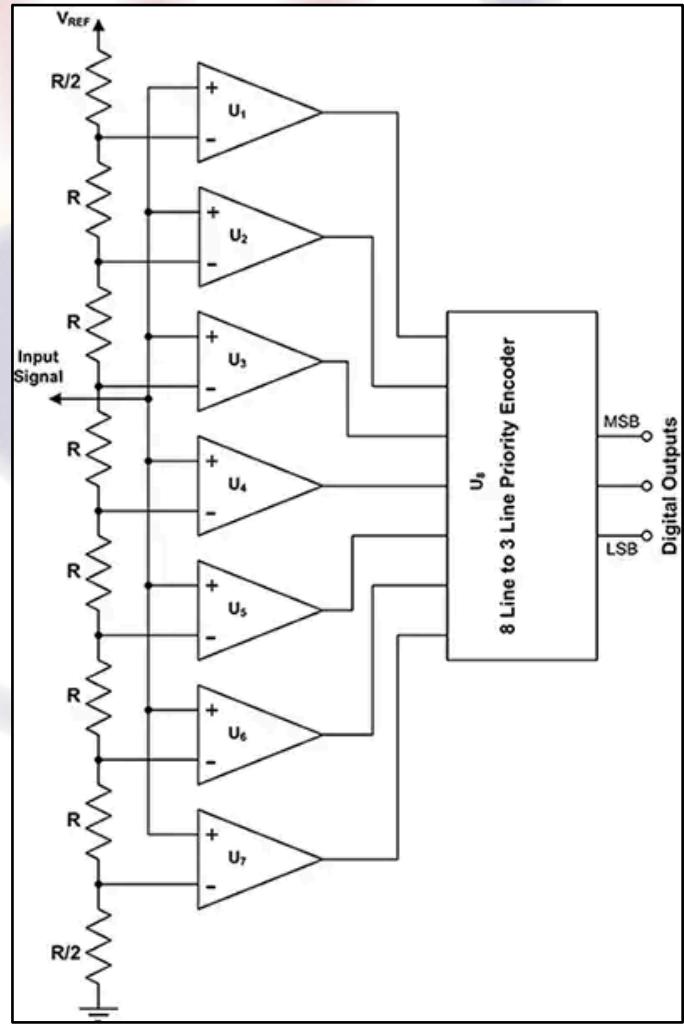
Tecnología de las cosas

Comunicaciones inter-placa (periféricos)

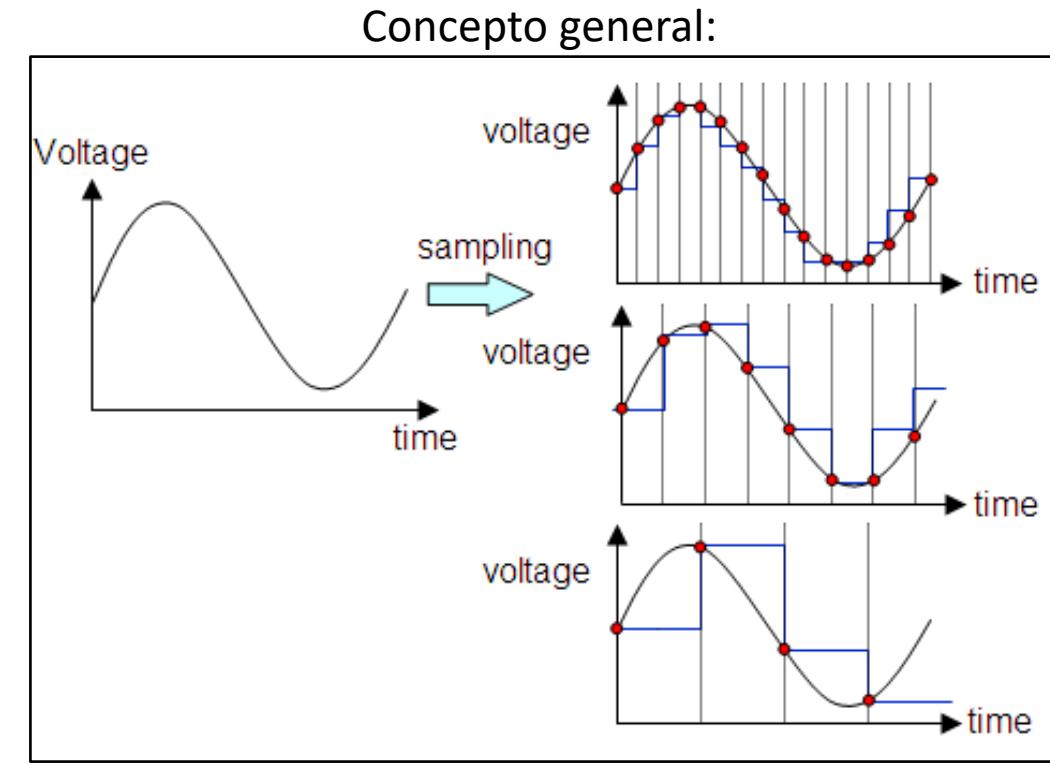


Tecnología de las cosas

Analog-Digital Converter (ADC)



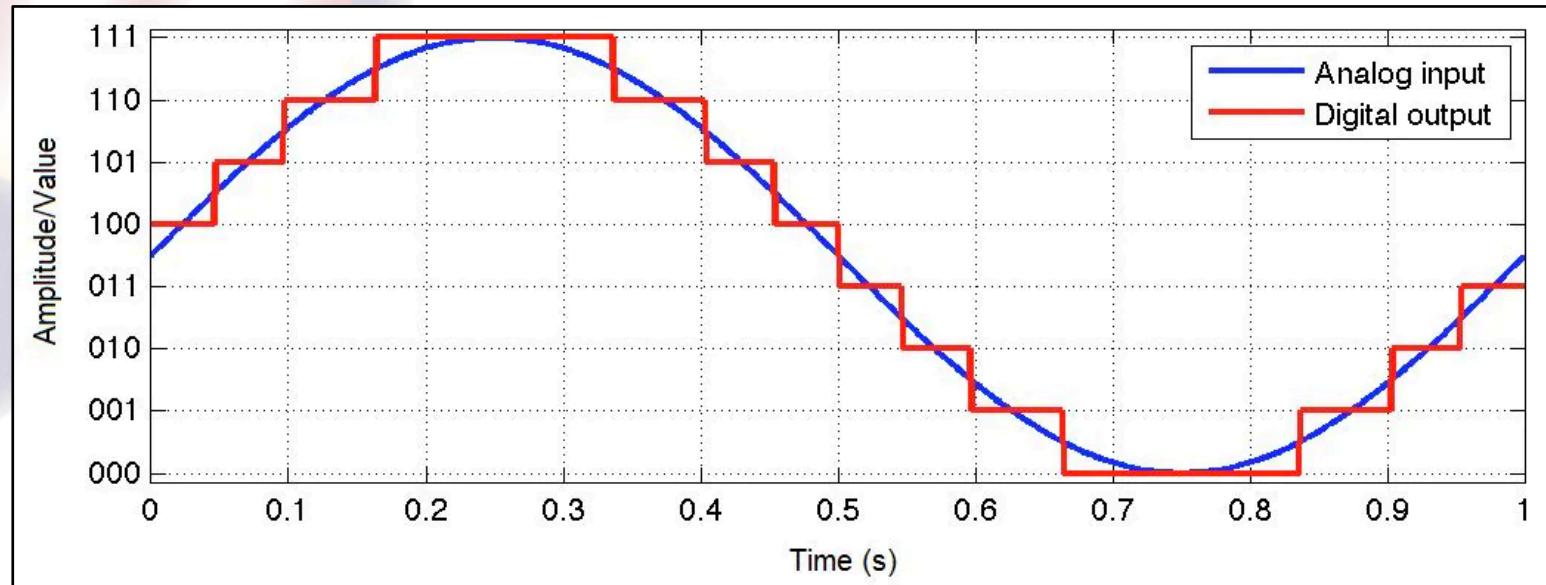
<https://www.digikey.com/es/articles/match-the-right-adc-to-the-application>



<https://www.flu-project.com/2019/09/teleco-in-nutshell-v70-conversion.html>

Tecnología de las cosas

Analog-Digital Converter (ADC)



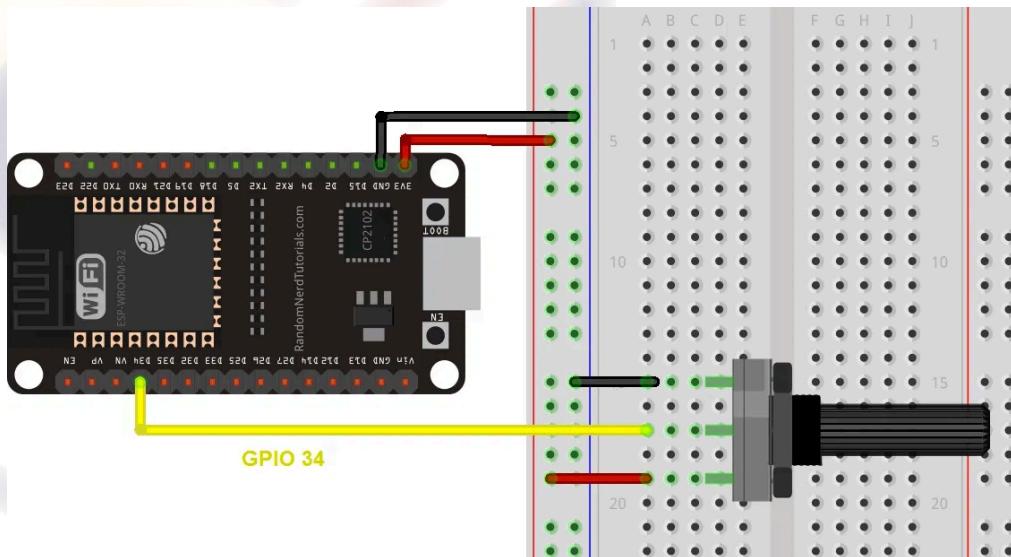
Características básicas

- Período de muestreo (sampling period) T_s
- Resolución (n. bits)
- Rango de entrada (Vref)

Tecnología de las cosas

Analog-Digital Converter (ADC)

Uso en la práctica



<https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/>

Ejemplo Arduino

```
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;

void setup() {
  Serial.begin(115200);
  delay(1000);
}

void loop() {
  // Reading potentiometer value
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(500);
}
```

Tecnología de las cosas

Analog-Digital Converter (ADC)

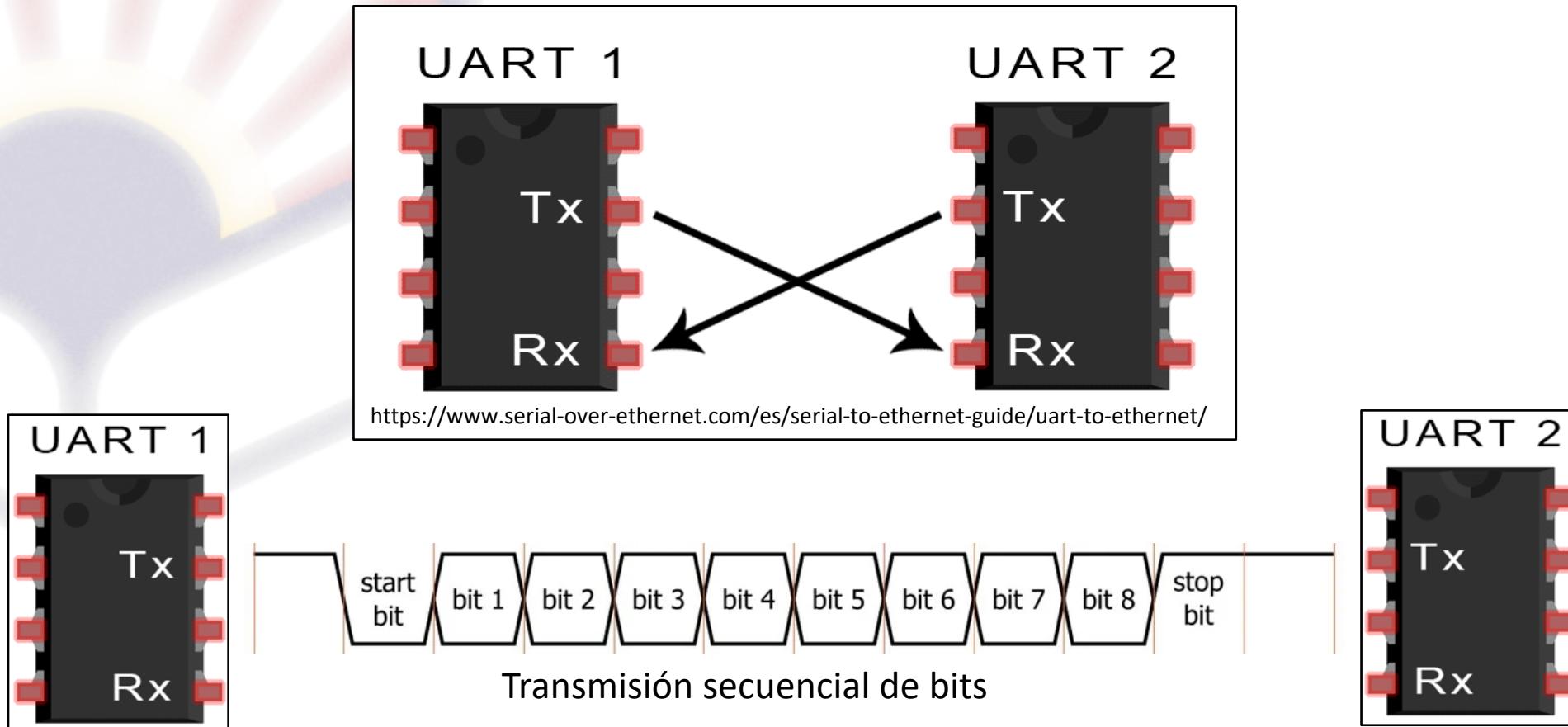
Uso en la práctica

- **OJO: Valores digitales crudos (LSB)**
 - Ejemplo de sensor (especificaciones): 0-1V -> -40-60ºC
 - Ejemplo de ADC: 10 bits -> 1024 valores digitales:
 - Regla de 3 simple:
 - 0 -> -40
 - 1024 -> 60
 - Calibración:
 - $y = mx + n$
 - $m = \text{inc.y} / \text{inc.x}$
 - $m = 100/1024$
 - $n = -40$
- float temperatura(uint_16 lsb):***
{
return (100.0/1024.0) * lsb - 40.0;
}
- ¿Error?**

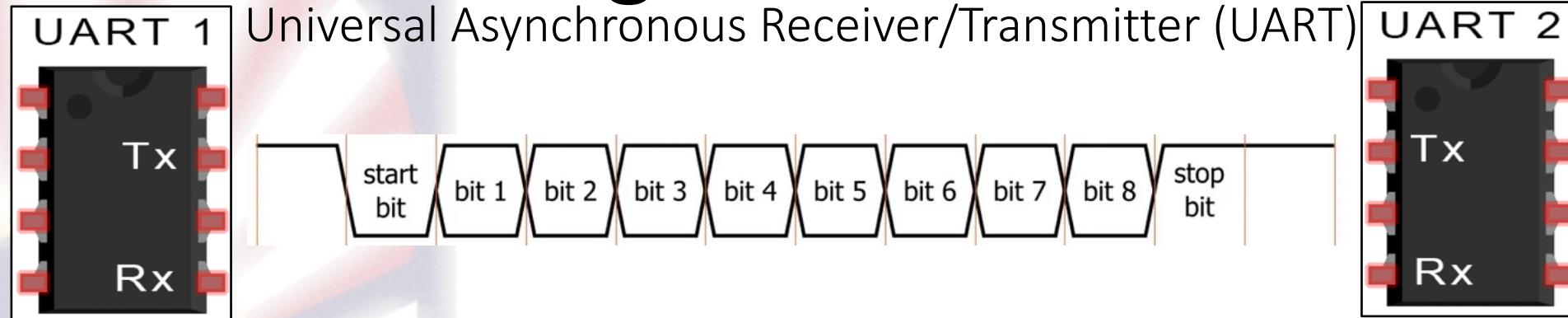
Tecnología de las cosas

Universal Asynchronous Receiver/Transmitter (UART)

Concepto general



Tecnología de las cosas



Características básicas

- Bit de inicio: marca el comienzo de la transferencia
- Bit de parada: marca el final de la transferencia
- Frecuencia?: a configurar en ambos dispositivos. *Baudios o bps (bits per second)*

Pregunta:

¿Por qué se llama asíncrono?

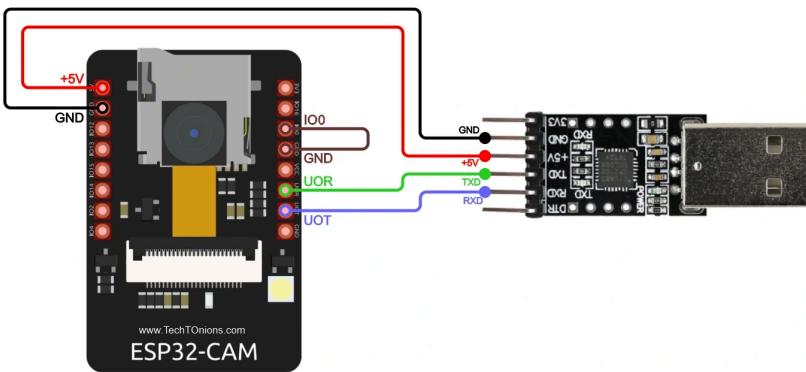
- No se comparte señal de reloj, cada equipo es responsable de monitorizar el puerto
- La comunicación, entendida de byte en byte, puede ocurrir en cualquier momento

Tecnología de las cosas

Universal Asynchronous Receiver/Transmitter (UART)

Uso en la práctica

Ejemplo micropython (ESP32)



<https://www.techtonions.com/esp32-cam-diy-programming-shield/>

```
from machine import UART
```

```
uart = UART(1, 9600)                      # init with given baudrate
uart.init(9600, bits=8, parity=None, stop=1) # init with given parameters
```

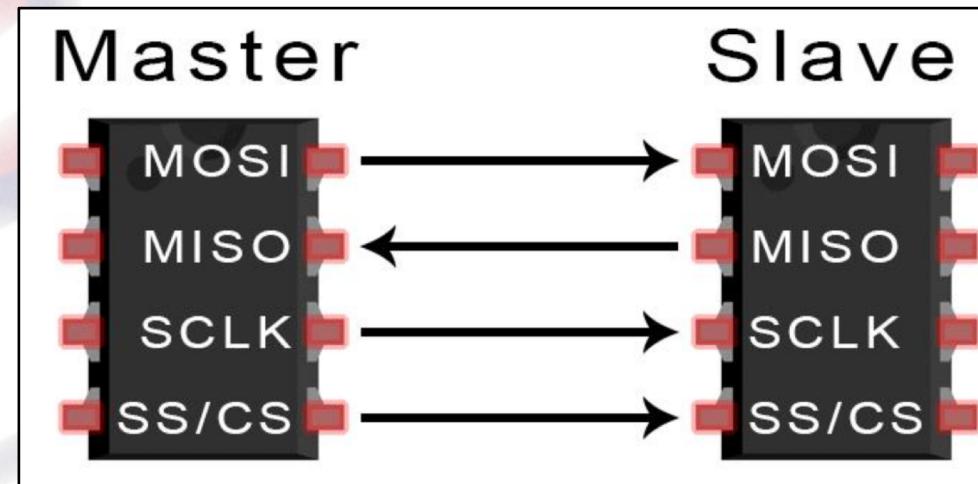
```
uart.read(10)      # read 10 characters, returns a bytes object
uart.read()        # read all available characters
uart.readline()   # read a line
uart.readinto(buf) # read and store into the given buffer
uart.write('abc')  # write the 3 characters
```

<https://docs.micropython.org/en/latest/library/machine.UART.html>

Tecnología de las cosas

Serial Peripheral Interface (SPI)

Concepto general



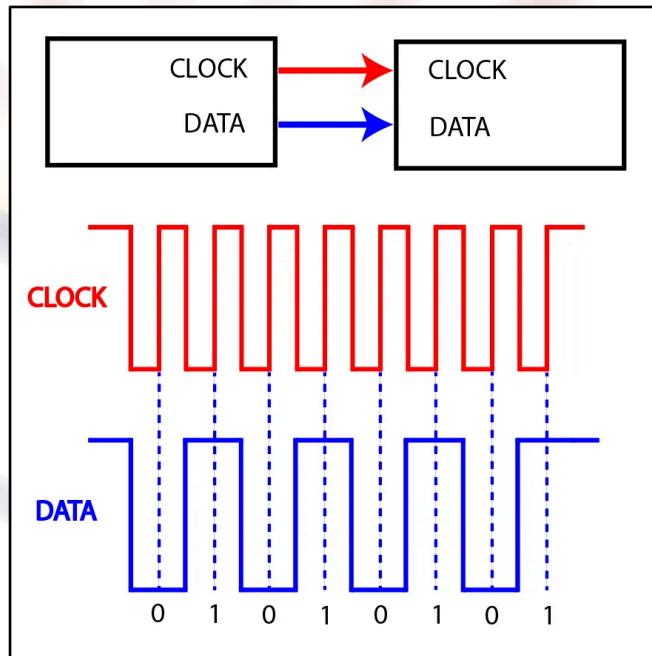
<https://atmega32-avr.com/basics-of-the-spi-communication-protocol/>

Señales:

- MOSI: Master Output Slave Input
- MISO: Master Input Slave Output
- SCLK: Señal de reloj (master)
- SS/CS: Slave Select / Chip Select

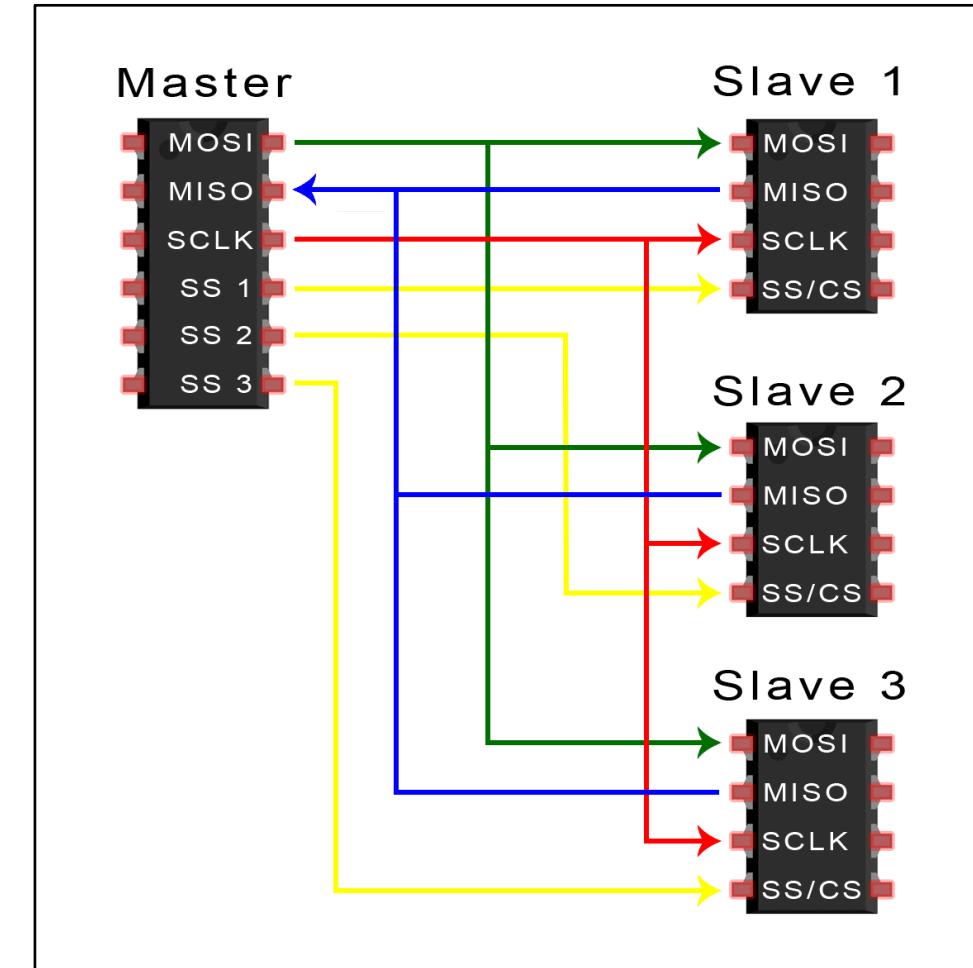
Tecnología de las cosas

Serial Peripheral Interface (SPI)



Características básicas

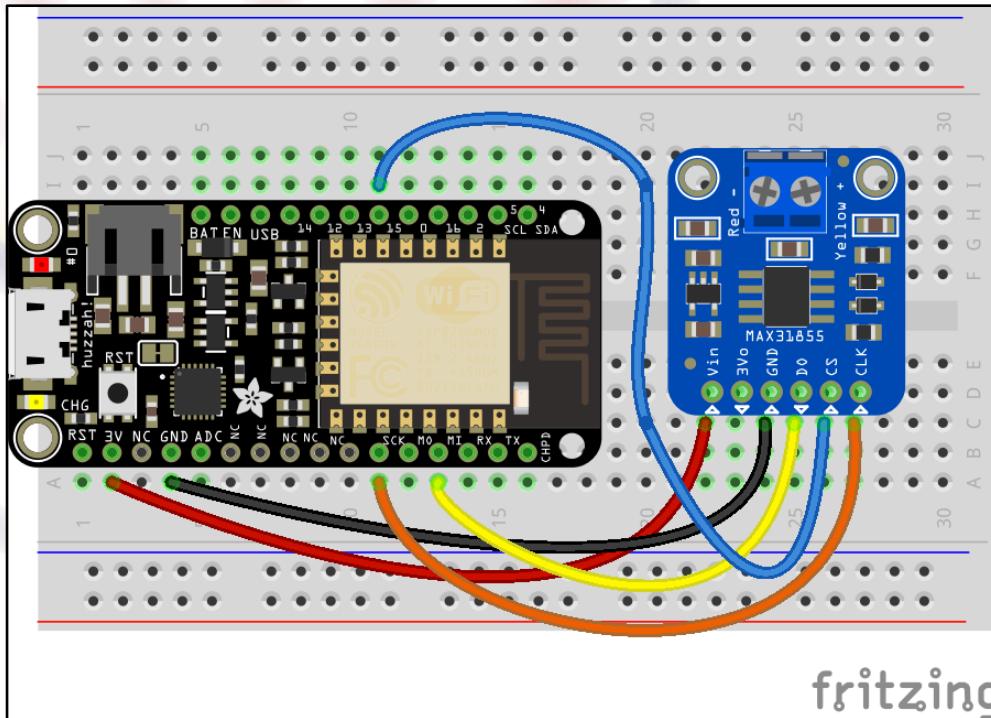
- Es un **bus de campo**
- Es **síncrono**
- Es **Full Duplex** (se puede enviar y recibir al mismo tiempo)
- Requiere arbitraje (hay un rol de maestro que tiene la iniciativa y controla el bus)



Tecnología de las cosas

Serial Peripheral Interface (SPI)

Uso en la práctica



<https://learn.adafruit.com/micropython-hardware-spi-devices>

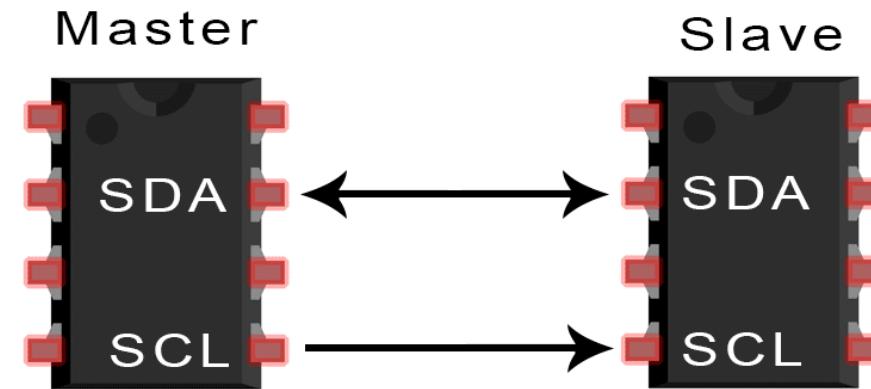
Ejemplo micropython (ESP32)

```
1 import machine
2 ...
3 spi = machine.SPI(1, baudrate=5000000, polarity=0, phase=0)
4 cs = machine.Pin(15, machine.Pin.OUT)
5 cs.high()
6 ...
7 data = bytearray(4)
8 cs.low()
9 spi.readinto(data)
10 cs.high()
11
12 def temp_c(data):
13     temp = data[0] << 8 | data[1]
14     if temp & 0x0001:
15         return float('NaN') # Fault reading data.
16     temp >= 2
17     if temp & 0x2000:
18         temp -= 16384 # Sign bit set, take 2's compliment.
19     return temp * 0.25
20
```

Tecnología de las cosas

Inter-Integrated Circuit (I2C)

Concepto general



<https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>

Señales:

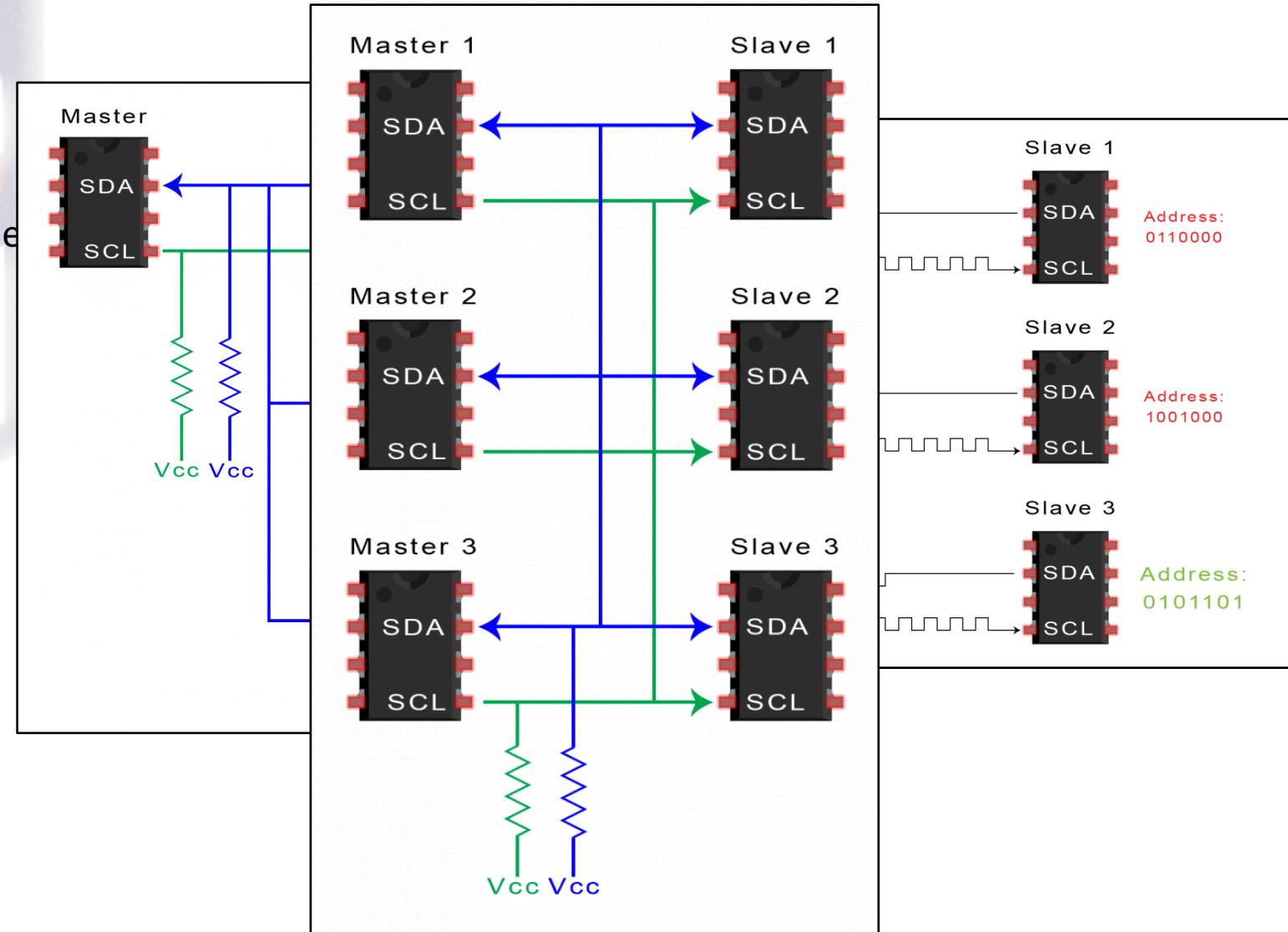
- SDA: Serial Data
- SCL: Serial Clock

Tecnología de las cosas

Inter-Integrated Circuit (I2C)

Características básicas

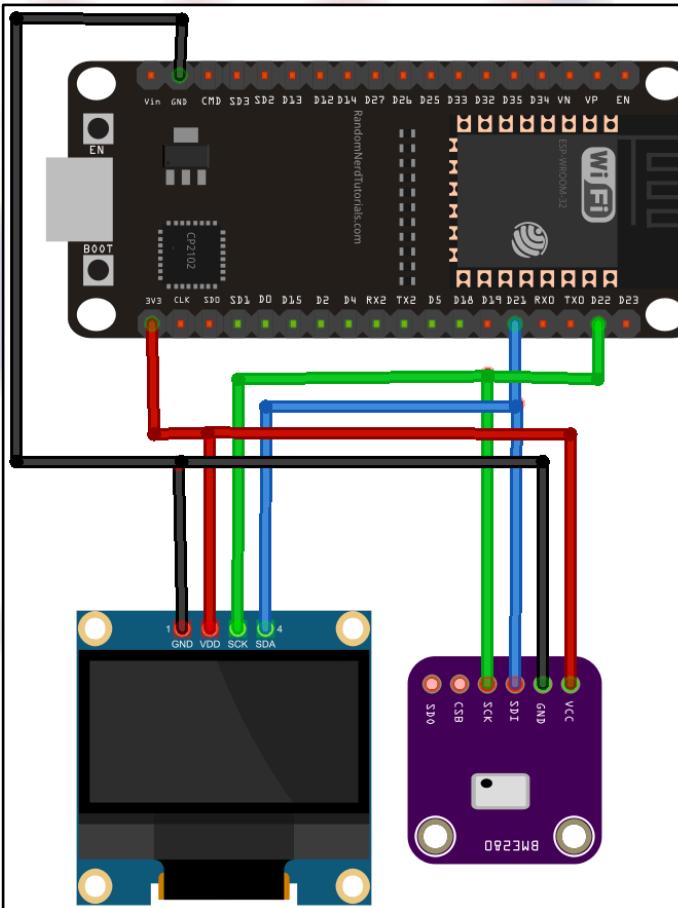
- Es un **bus de campo**
- Es **síncrono**
- Es **Half Duplex** (no se puede enviar y recibir al mismo tiempo)
- Requiere arbitraje (hay un rol de maestro que tiene la iniciativa y controla el bus)
- Solo tiene 2 hilos (implementa direcciones digitales)
- Permite múltiples nodos máster.
- Es más complejo de implementar y más lento que SPI.



Tecnología de las cosas

Inter-Integrated Circuit (I2C)

Uso en la práctica



Ejemplo micropython (ESP32)

```
from machine import I2C  
  
i2c = I2C(freq=400000)  
  
# create I2C peripheral at frequency of 400kHz  
# depending on the port, extra parameters may be required  
# to select the peripheral and/or pins to use  
  
i2c.scan()  
  
# scan for peripherals, returning a list of 7-bit addresses  
  
i2c.writeto(42, b'123')  
i2c.readfrom(42, 4)  
  
# write 3 bytes to peripheral with 7-bit address 42  
# read 4 bytes from peripheral with 7-bit address 42  
  
i2c.readfrom_mem(42, 8, 3)  
# read 3 bytes from memory of peripheral 42,  
# starting at memory-address 8 in the peripheral  
i2c.writeto_mem(42, 2, b'\x10') # write 1 byte to memory of peripheral 42  
# starting at address 2 in the peripheral
```

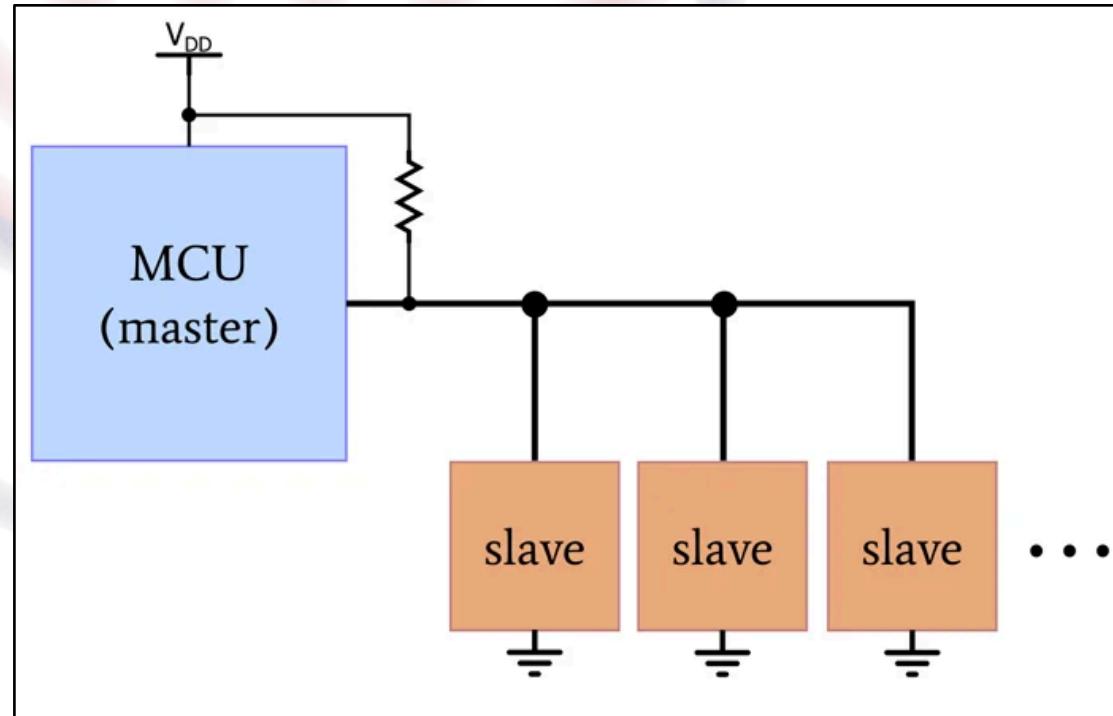
<https://docs.micropython.org/en/latest/library/machine.I2C.html>

<https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/>

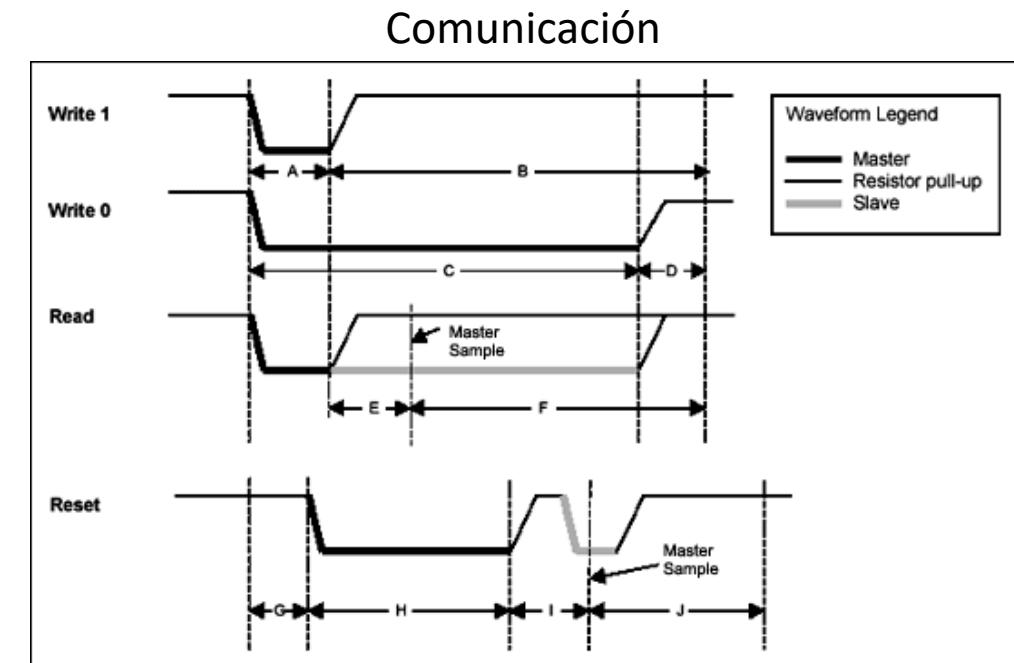
Tecnología de las cosas

OneWire

Concepto general



<https://www.allaboutcircuits.com/technical-articles/low-pin-count-serial-communication-introduction-to-the-1-wire-bus/>



<https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html>

Tecnología de las cosas

OneWire

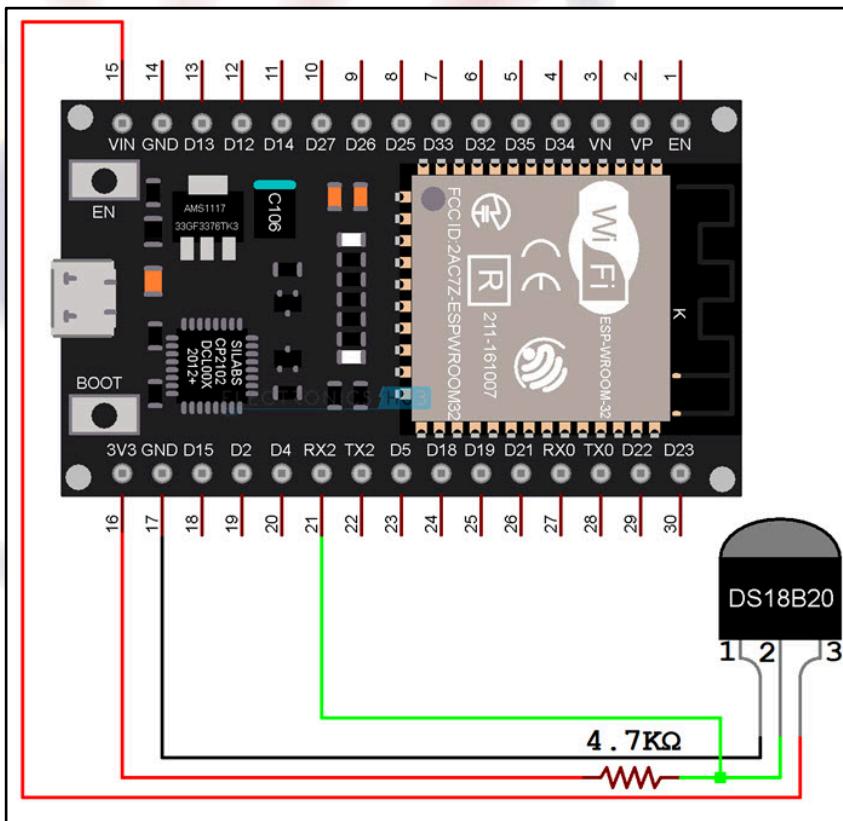
Características básicas

- Es un bus de campo
- Es **síncrono**
- Es **Half Duplex** (no se puede enviar y recibir al mismo tiempo)
- Requiere arbitraje (hay un rol de maestro que tiene la iniciativa y controla el bus)
- Solo tiene 1 hilos (implementa direcciones digitales)
- Cada dispositivo esclavo tiene una dirección única e inalterable de 64 bits.
- Más lento que SPI e I2C
- Consumo muy bajo
- Distancias relativamente largas
- Comunicación a 3 fases:
 - Sincronización
 - Identificación
 - Comando a memoria

Tecnología de las cosas

OneWire

Uso en la práctica



Ejemplo micropython (ESP32)

```
import time
import machine
import onewire, ds18x20

# the device is on GPIO12
dat = machine.Pin(12)

# create the onewire object
ds = ds18x20.DS18X20(onewire.OneWire(dat))

# scan for devices on the bus
roms = ds.scan()
print('found devices:', roms)

# loop 10 times and print all temperatures
for i in range(10):
    print('temperatures:', end=' ')
    ds.convert_temp()
    time.sleep_ms(750)
    for rom in roms:
        print(ds.read_temp(rom), end=' ')
    print()
```

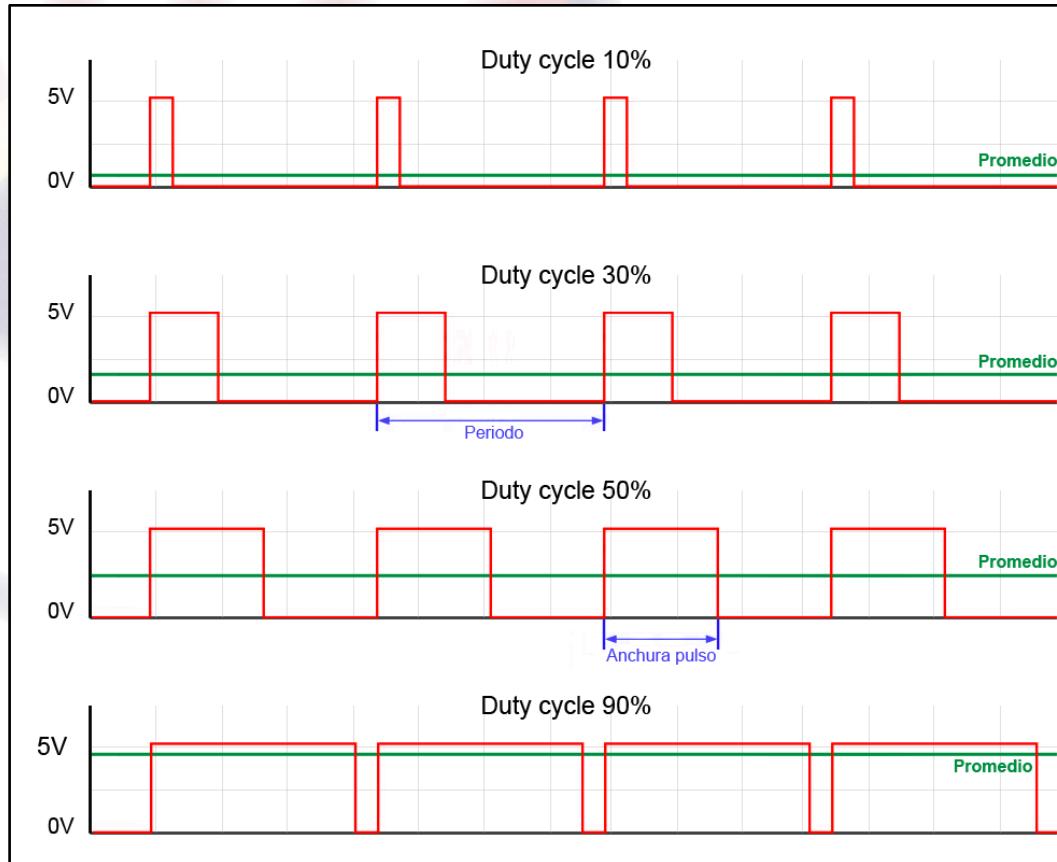
<https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/>

<https://docs.micropython.org/en/latest/esp8266/tutorial/onewire.html>

Tecnología de las cosas

Pulse-Width Modulation (PWM)

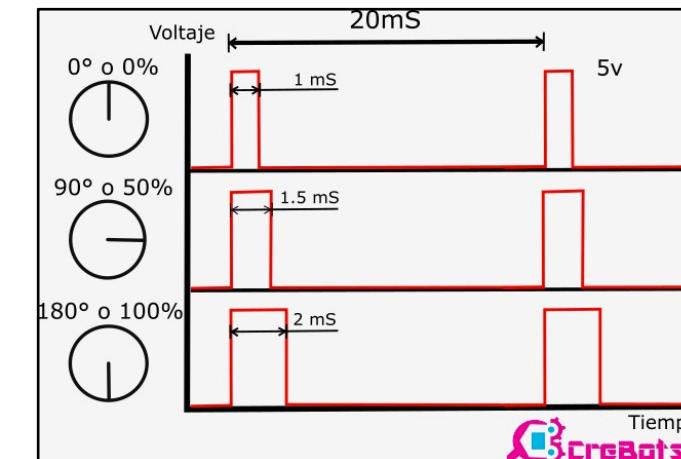
Concepto general



<https://www.luisllamas.es/salidas-analogicas-pwm-en-arduino/>

Se trata de expresar una proporción (concepto analógico) mediante una señal digital.

Es un potenciómetro digitalizado



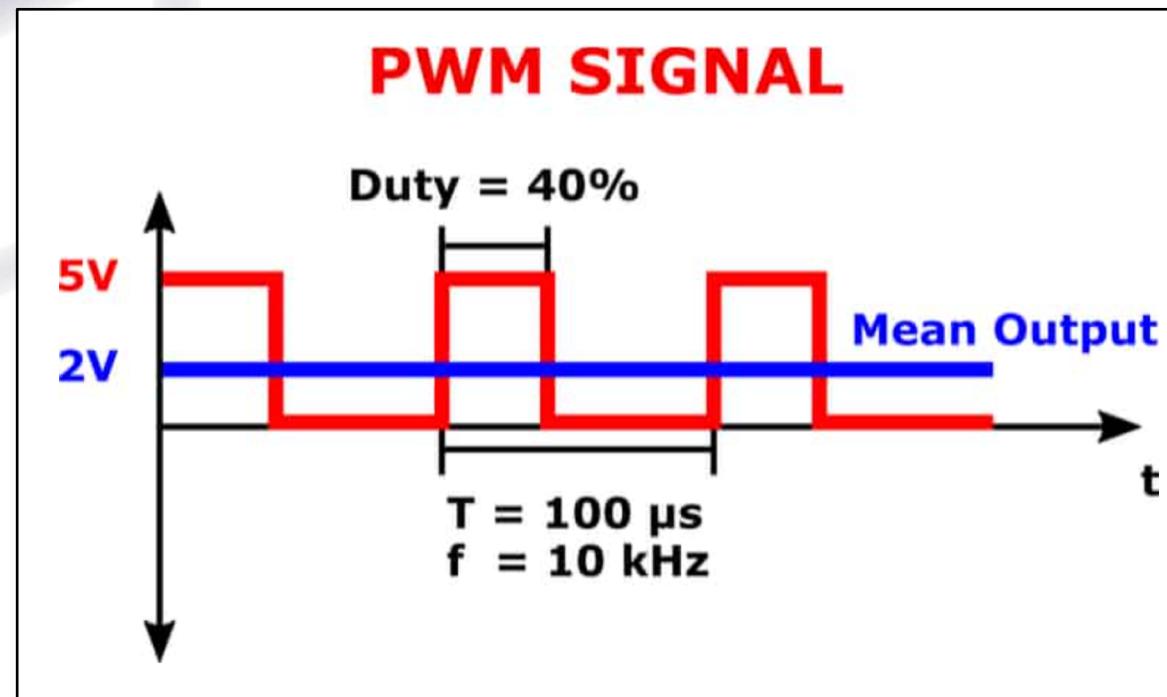
<https://crebots.com/protocolosrc/>

Tecnología de las cosas

Pulse-Width Modulation (PWM)

Características básicas

- Es extremadamente **simple de implementar**
- Dos parámetros: **ciclo de trabajo** y **frecuencia**
- Se utiliza para **controlar parámetros analógicos** (motores DC, Leds, etc.)

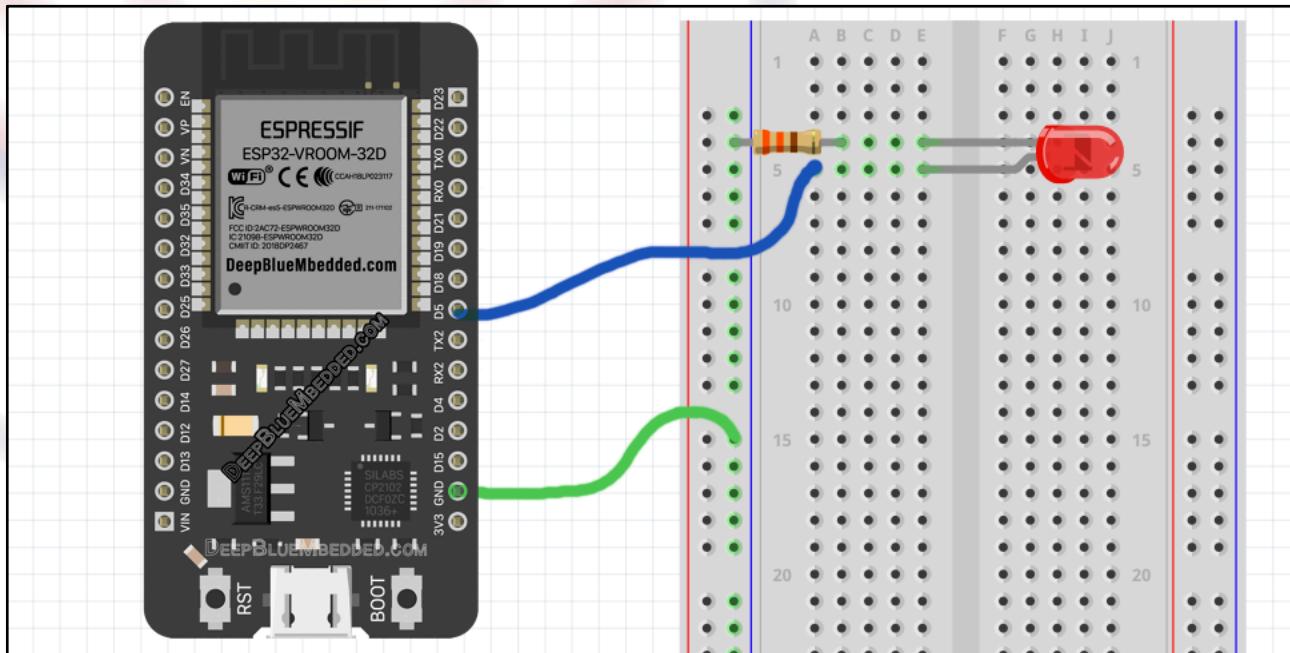


<https://resourcespcb.cadence.com/blog/2020-pulse-width-modulation-characteristics-and-the-effects-of-frequency-and-duty-cycle>

Tecnología de las cosas

Pulse-Width Modulation (PWM)

Uso en la práctica



<https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/>

Ejemplo Arduino IDE (ESP32)

```
#define LED_GPIO      5
#define PWM1_Ch       0
#define PWM1_Res      8
#define PWM1_Freq     1000

int PWM1_DutyCycle = 0;

void setup()
{
    ledcAttachPin(LED_GPIO, PWM1_Ch);
    ledcSetup(PWM1_Ch, PWM1_Freq, PWM1_Res);
}

void loop()
{
    while(PWM1_DutyCycle < 255)
    {
        ledcWrite(PWM1_Ch, PWM1_DutyCycle++);
        delay(10);
    }
    while(PWM1_DutyCycle > 0)
    {
        ledcWrite(PWM1_Ch, PWM1_DutyCycle--);
        delay(10);
    }
}
```

Tecnología de las cosas

¿Preguntas?