UNCUYO
UNIVERSIDAD NACIONAL DE CUYO
FACULTAD DE INGENIERÍA
Licenciatura en Ciencias de la Computación

Sistemas Embebidos

Unidad 1

Introducción a los sistemas embebidos

Definición de sistema embebido, muchas…

"Dispositivo que contiene componentes de hardware y software fuertemente acoplados para realizar una única función: formar parte de grandes sistemas. No está destinado a ser programado de forma independiente por los usuarios. Se espera que trabaje con mínima o sin interacción humana. Dos características adicionales: operación reactiva y fuertemente restringido". (John Davies, "MSP430 Microcontroller Basics")

"Sistemas de procesamiento embebidos en productos del entorno de las personas o ambiente (autos, trenes, aviones, equipos de comunicaciones o industriales, juguetes, electrodomésticos, etc.)" (Marwedel)

"Computadora de pequeño tamaño que es parte de una máquina o un gran sistema eléctrico o mecánico. Es llamado embebido porque está embebido o incrustado dentro de otro aparato" (Perry Xiao).

Diagrama en bloques general de un sistema embebido

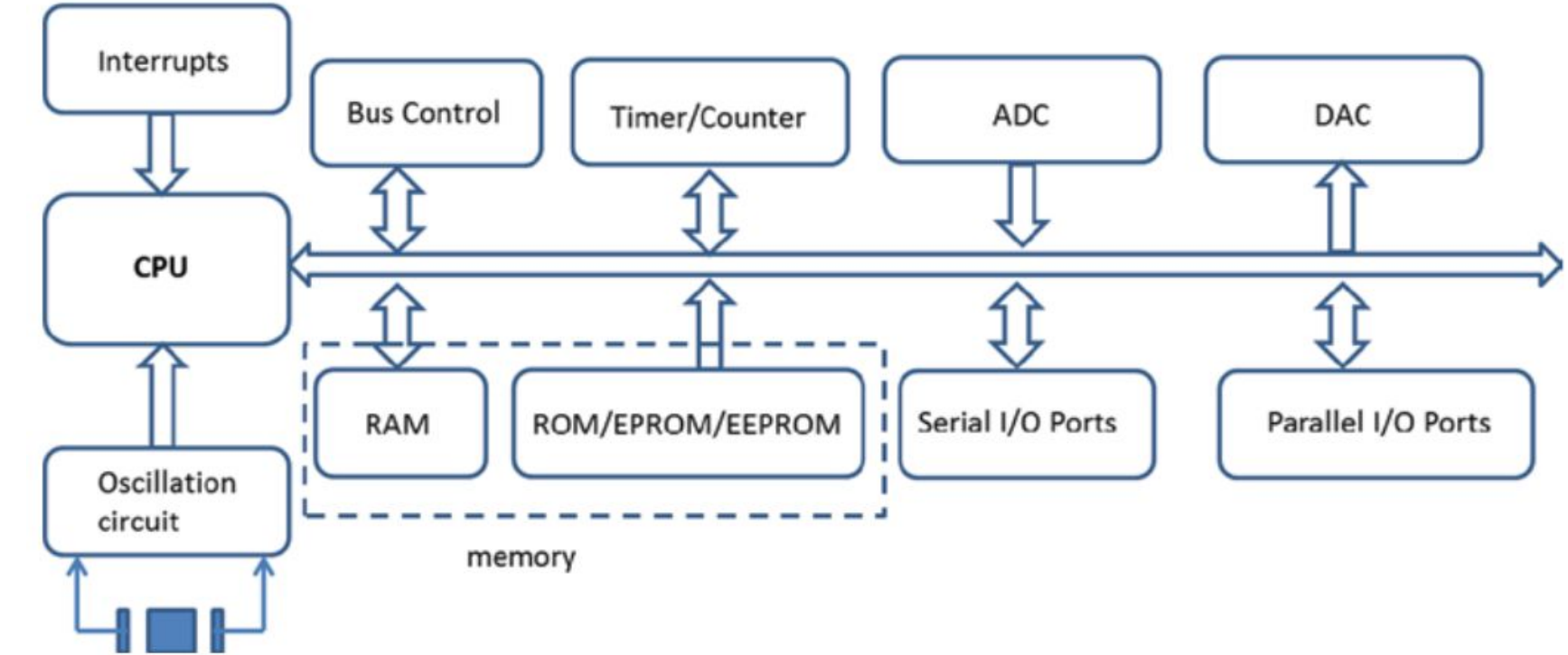
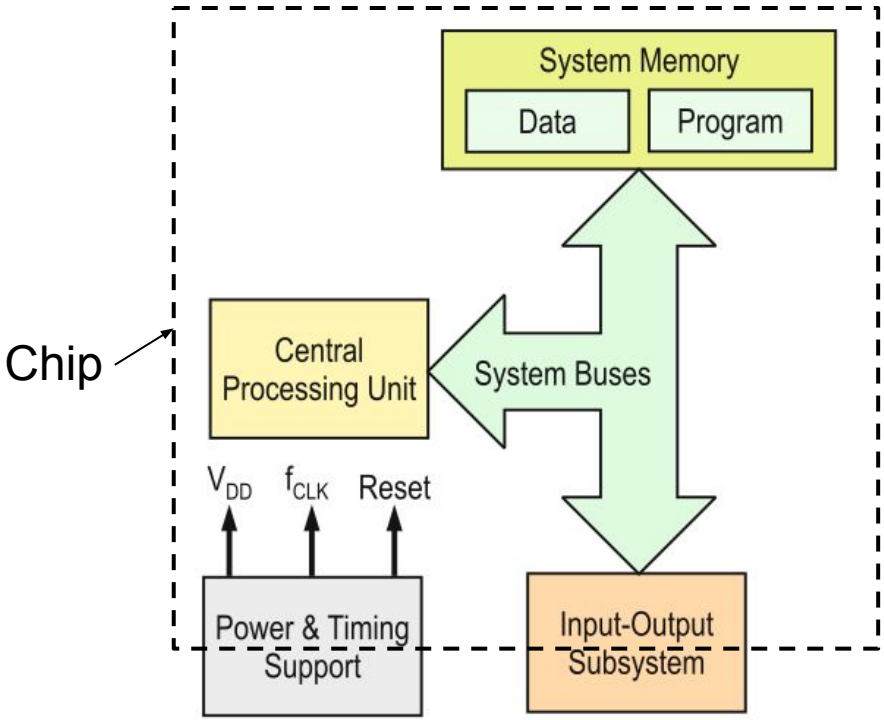


Figura obtenida de Perry Xiao ,"Designing Embedded Systems and the Internet of Things (IoT) with the ARM Mbed". Pag. 5

Microcontrolador



microcontrolador

Diferencias entre procesadores y microcontroladores:

	Microcontroladores	Computadoras
Integración	Todo en un chip (procesador, memoria, periféricos, etc.)	Componentes separados
Evolución centrada en	Reducción consumo, aplicaciones específicas.	Poder de procesamiento
SO	Usado en algunas aplicaciones (basadas en tiempo real), pero otras no	Obligatorio
Tamaño memorias	De pocos Kbytes a MBytes	Lo mayor posible (GBytes)

Sistemas embebidos e IoT

- **IoT o Internet de las cosas**: "*Presencia penetrante de variados dispositivos que contienen sensores y actuadores con un único esquema de direccionamiento, que son capaces de interactuar y cooperar unos con otros para lograr objetivos comunes*"
- Requisitos de dispositivos IoT:
  - Cada dispositivo debe tener una dirección única: solución IPv6 y 6LoWPAN.
  - Cada dispositivo debe poder comunicarse (Wifi, 6LoWPAN, NFC, RFID, ZigBee, USB, etc.)
  - Cada dispositivo debe tener sensores y actuadores. } SE
  - Cada dispositivo necesita un microcontrolador.
  - Plataformas de visualización, análisis y almacenamiento de datos.

Distribución de procesadores según bits en años recientes

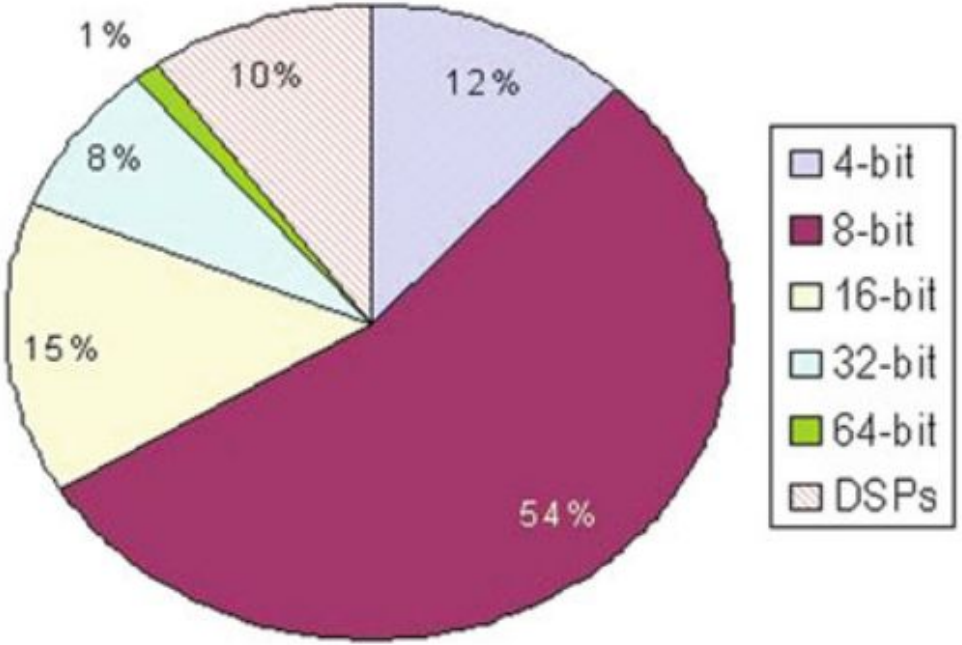


Figura obtenida de Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier, “Introduction to Embedded Systems Using Microcontrollers and the MSP430”. Página. 5.

Clasificación según número de bits:

- Gama baja: 8 bits, 4 bits:
  - Periféricos específicos (sensores y motores). Conectados a pines.
  - Comunicación mediante protocolos específicos de bajo costo.
  - Aparatos muy simples, tostadora, lavarropas simples, máquinas de hornear pan, etc.
- Gama media: 16 bits:
  - Electrodomésticos con más funcionalidades (incluyen WiFi).
- Gama alta: 32 o más bits:
  - No muy diferentes a una computadora de propósito general + periféricos específicos de sistemas embebidos.
  - Comunicación a través de TCP/IP, Ethernet, Wifi.
  - Routers, equipos de red, SmartTVs, Raspberry.

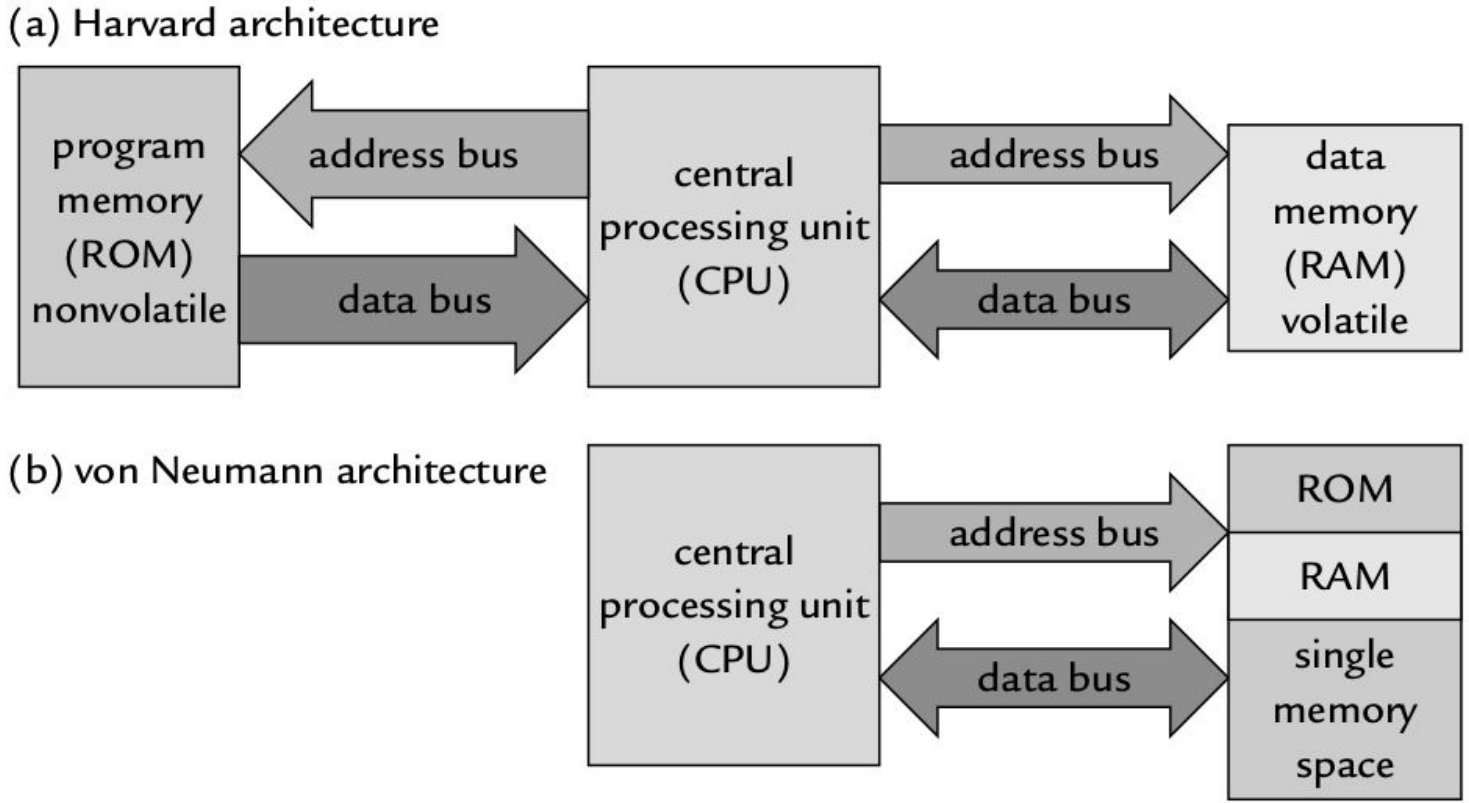


Figura obtenida de: John Davies, "MSP430 Microcontroller Basics", pag. 14

Aplicaciones más importantes de los sistemas embebidos

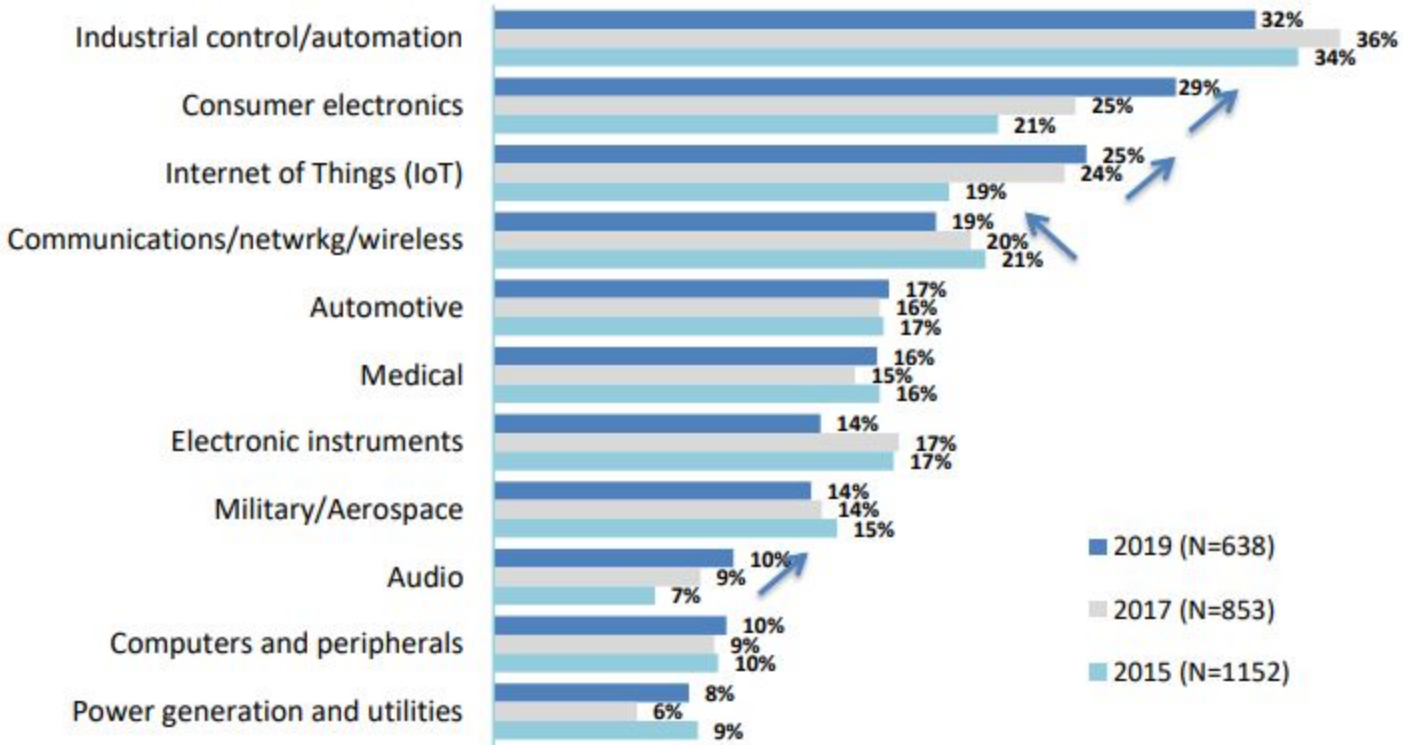


Figura obtenida de EETimes, “2019 Embedded Markets Study”.
https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf

Lenguajes más utilizados para programar sistemas embebidos

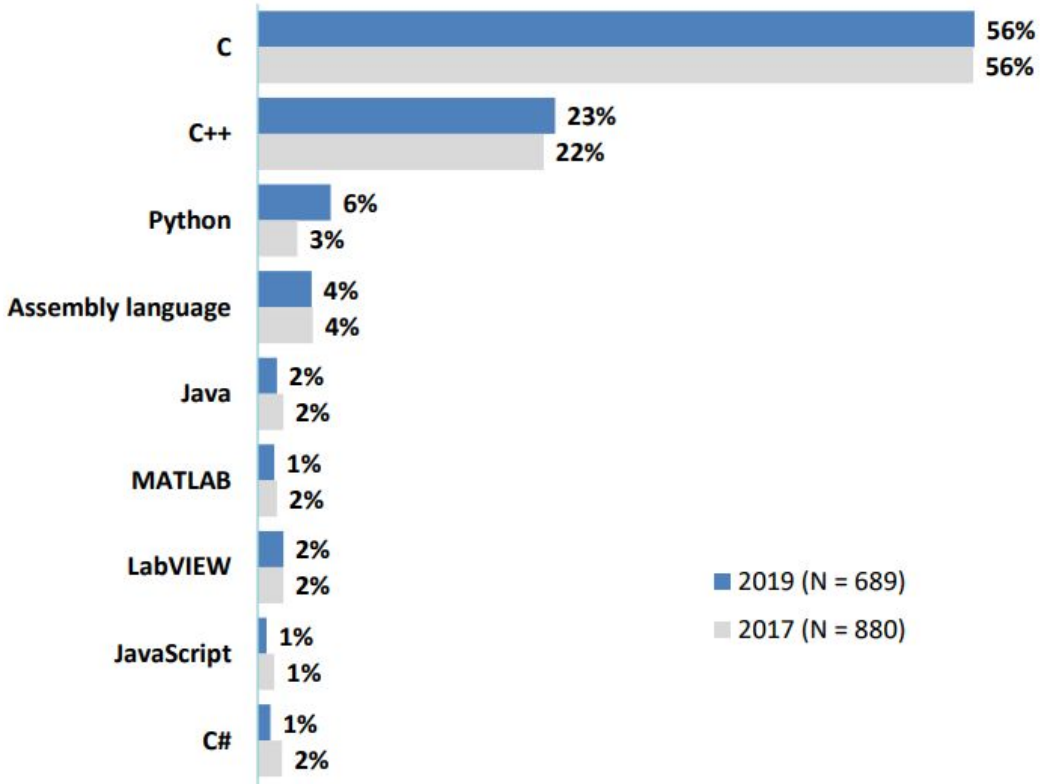


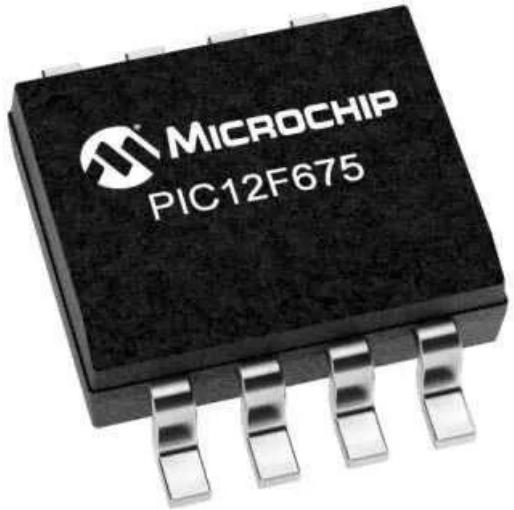
Figura obtenida de EETimes, “2019 Embedded Markets Study”.
https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf

Familia PIC

- Creados por Microchip Technology Inc.
- Arquitectura Harvard.
- Programable en lenguaje ensamblador.
  - Interfaces de desarrollo en lenguaje C (MPLAB).

8 bits (PIC 12F675).

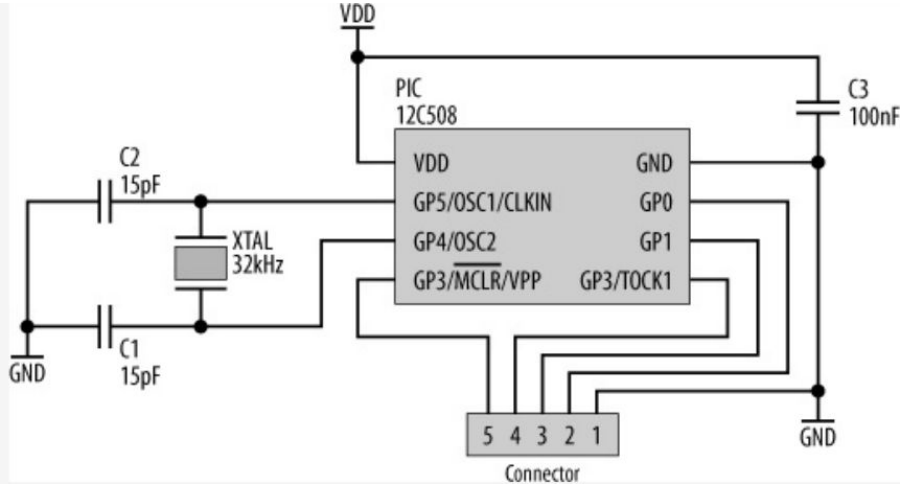
- Memoria de programa de 512 palabras.
- RAM de 25 bytes.
- Comunicación: I2C, SPI.
- Salidas: digitales y PWM.
- Frecuencias de operación:
  - 4 MHz
  - 32 KHz (bajo consumo)



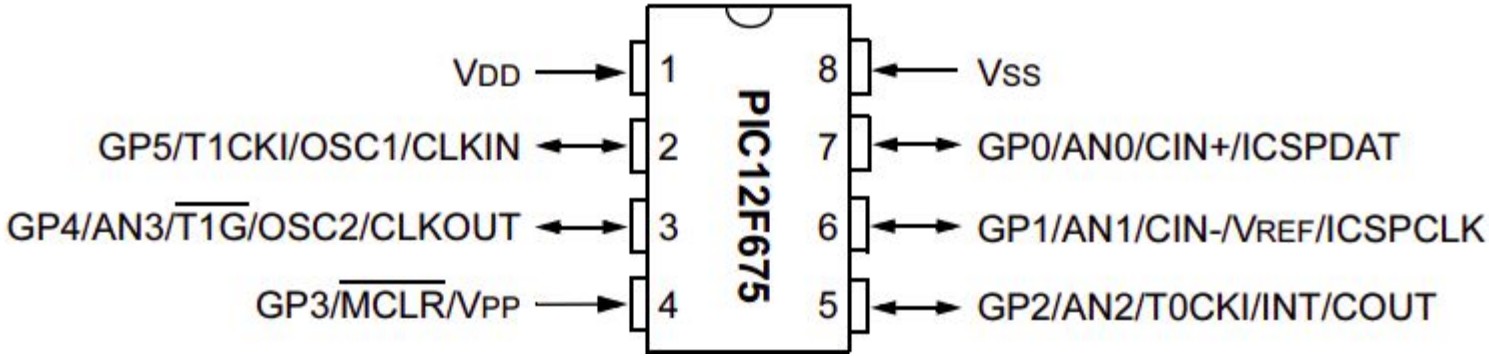
Familia PIC

8 bits (PIC 12F675), continuación.

- ALU con registro acumulador (W).
- 35 instrucciones.
- Consumo:
  - Normal: 100µA (625 días con pilas AA).
  - Bajo consumo: 1 nA (171mil años con pilas AA).
- Herramientas desarrollo en C o ensamblador.
- Precio: 1.60 USD (Ebay)



Familia PIC:8 bits (PIC 12F675).



GP0: Entrada salida digital
T1CKI: Timer 1 clock imput
AN0: Entrada analógica 0
CIN+ y CIN-: Comparadores analógicos
COUT: Salida comparador
ICSPDAT: Comunicación serial

Figura obtenida de Microchip, PIC12F629/675 Data Sheet.

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYC

s de la

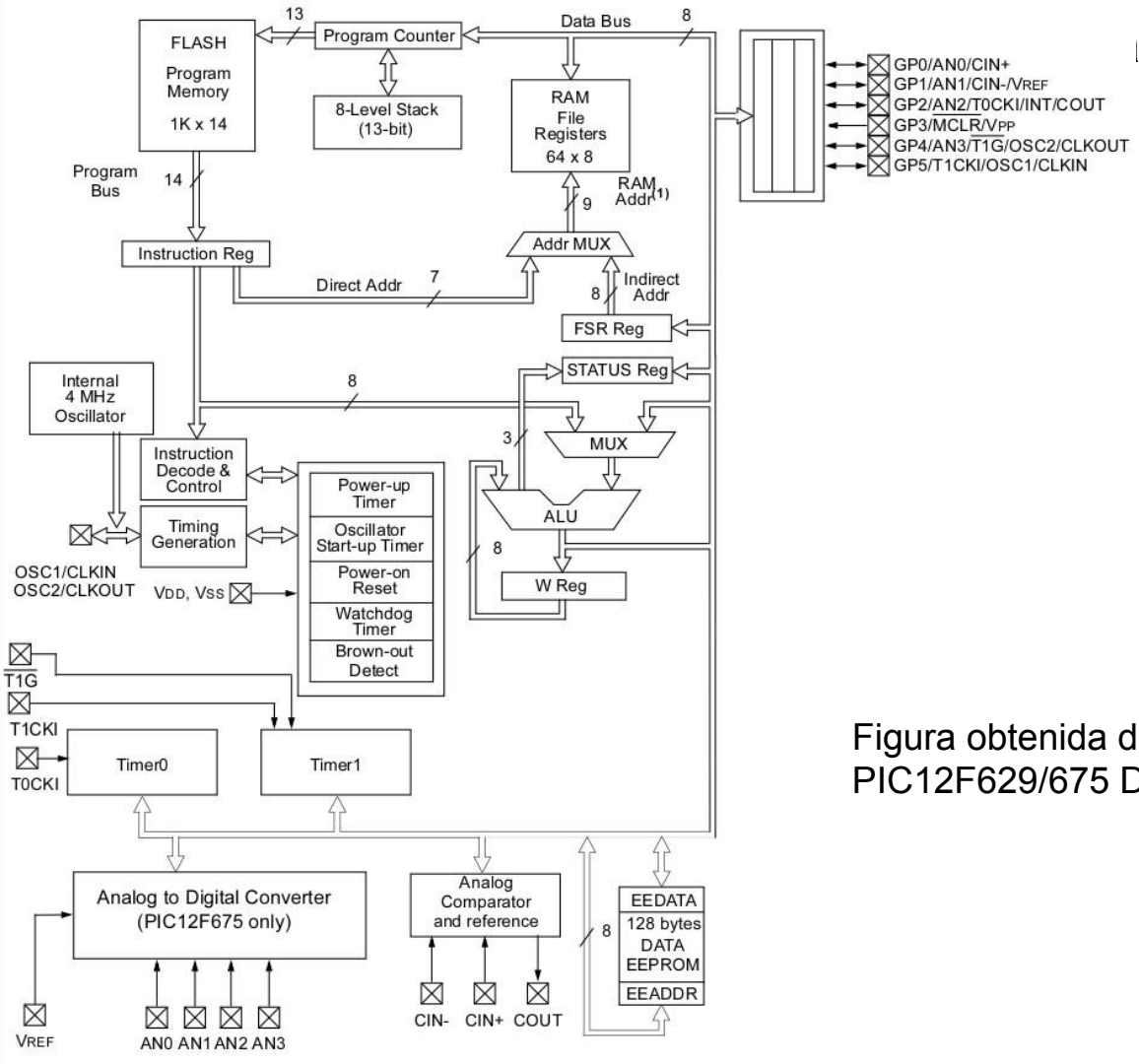


Figura obtenida de Microchip, PIC12F629/675 Data Sheet.

Memoria RAM y mapeo en memoria

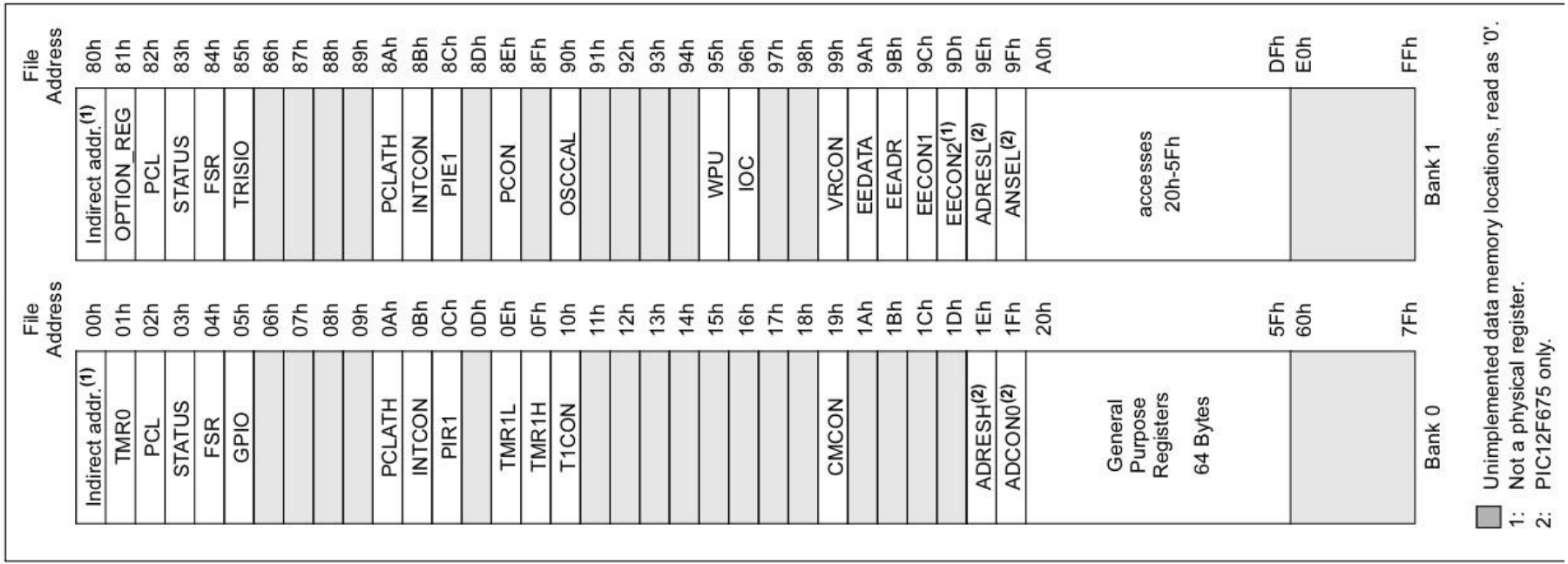
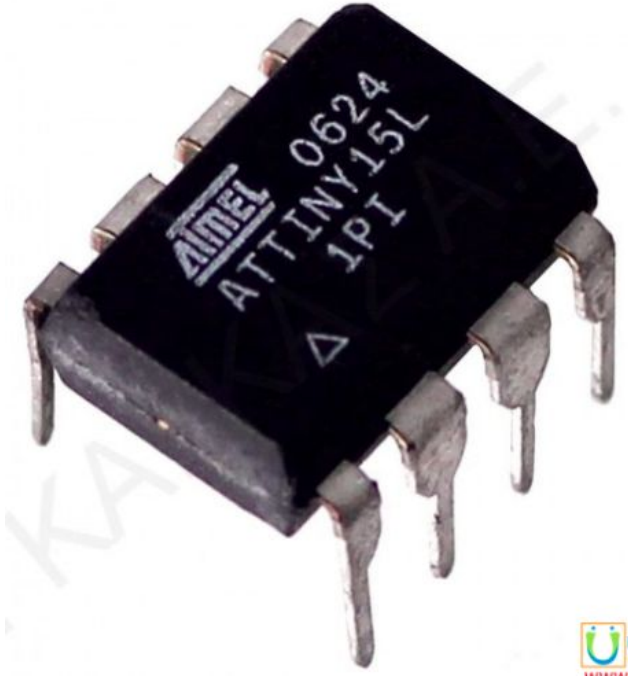


Figura obtenida de Microchip, PIC12F629/675 Data Sheet.

Familia AVR

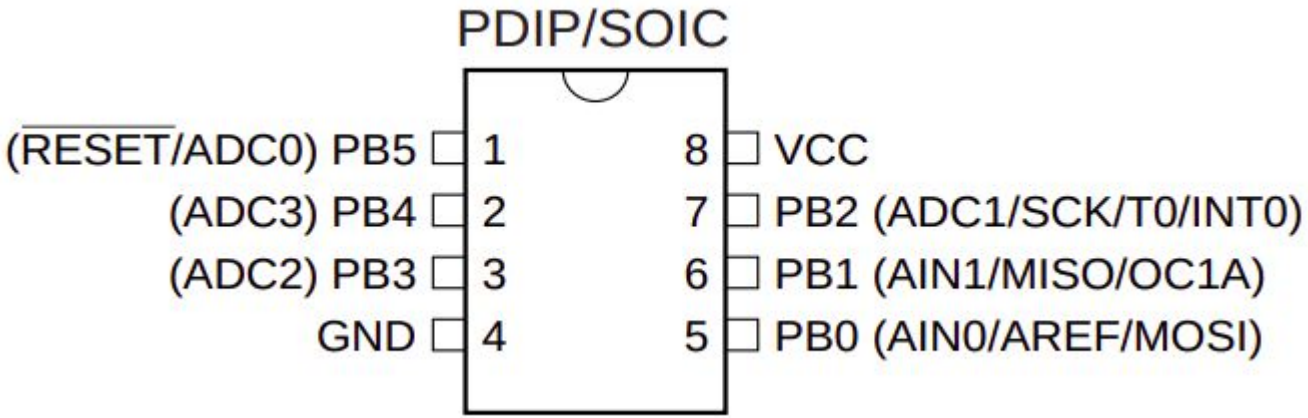
Procesador ATtiny15

- 8 bits.
- Memoria programa: 512 palabras
- 90 Instrucciones
- Memoria RAM datos: No posee
- 32 registros.
- EEPROM: 64 bytes
- 5 pines de I/O.
- Conversores A/D
- Precio: 1.60 USD (Ebay)



Familia AVR

Procesador ATtiny15



INT0: Interrupción externa

Familia AVR

Procesador ATmega328 (16 bits)

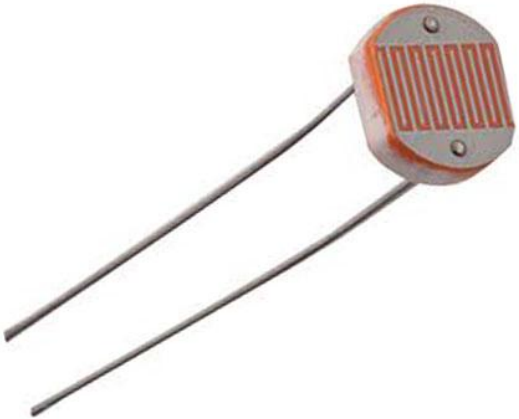
- Memoria programa: 32KB
- 131 interrupciones
- Memoria RAM datos: 2 KB
- Memoria EEPROM: 1 KB
- frecuencia: 16 MHz
- Conversores AD y DA, SPI, I2C, USAR.
- Consumo:
  - Normal: 1.5 mA (41 días con pilas AA)
  - Bajo consumo: 1 µA (171 años).
- Precio: 5.0 USD (Ebay)

JTAG

- Joint Test Action Group: Norma IEEE 1149.1
- Provee acceso al procesador y sus periféricos.
- Puerto que permite:
  - Escribir las memorias de programa.
  - In-circuit debug.
  - Examinar y escribir registros.
- Puerto de 4 pines.

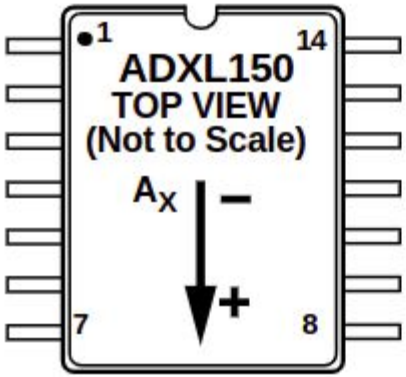
Sensores

LDR: Light Dependent Resistor)

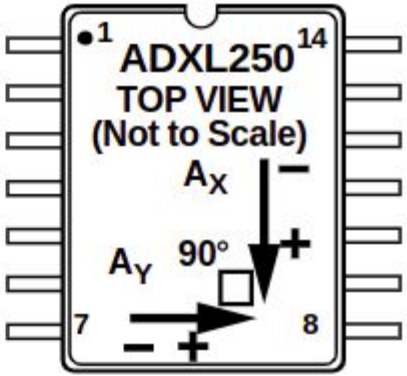


LM35: voltaje dependiente de la temperatura

Acelerómetro



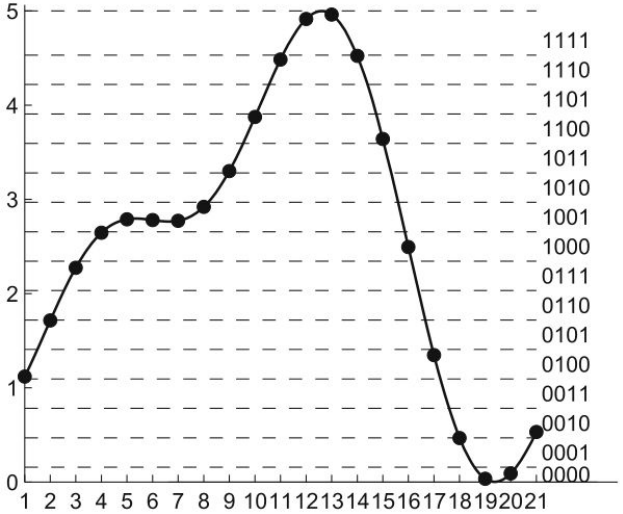
POSITIVE A = POSITIVE V_{OUT}

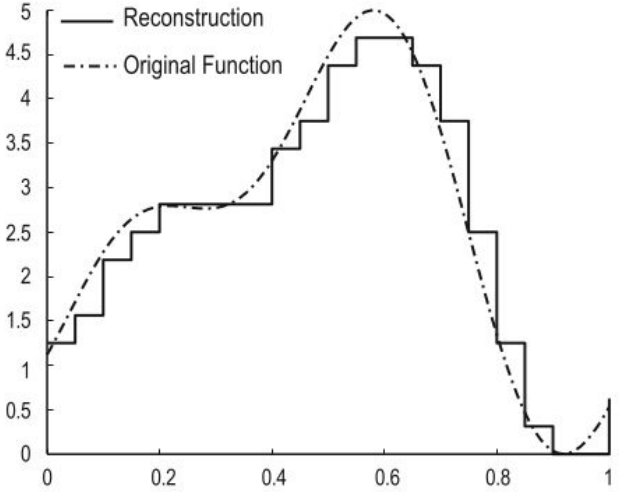


POSITIVE A = POSITIVE V_{OUT}

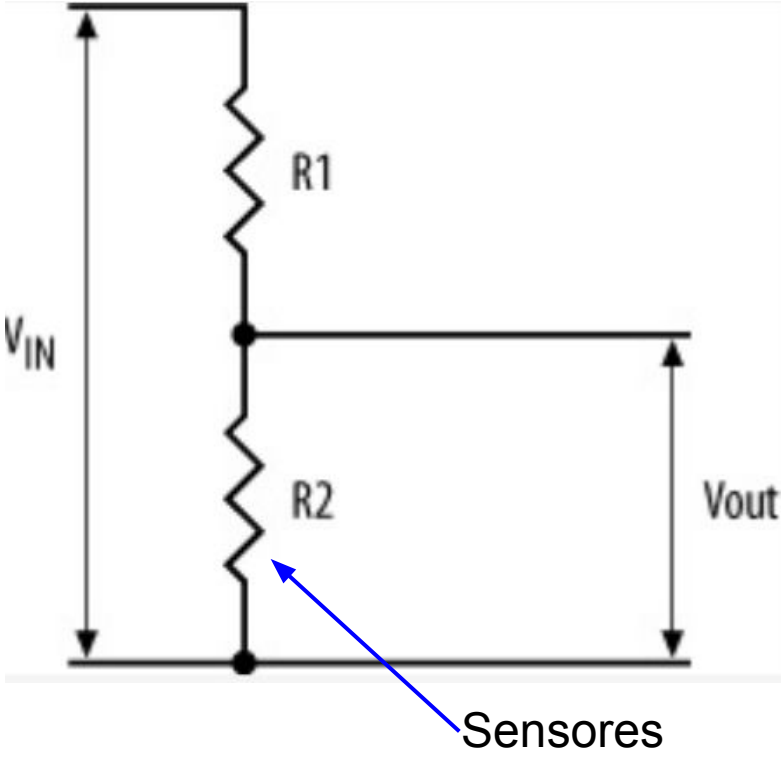
$$V_{OUT} = V_S/2 - (\text{sensitivity} * V_S/5 * \text{acceleration})$$

Micrófono (R variable)





Divisor de tensión



$$V_{OUT} = V_{IN} * R2 / (R1 + R2)$$

Ejemplo:

$$V_{OUT} = 5V \quad 1k / (1k + 1k)$$
$$= 5V \quad 1k / 2k$$
$$= 5V * 0.5$$
$$= 2.5V$$

Sistemas Embebidos

Unidad 2

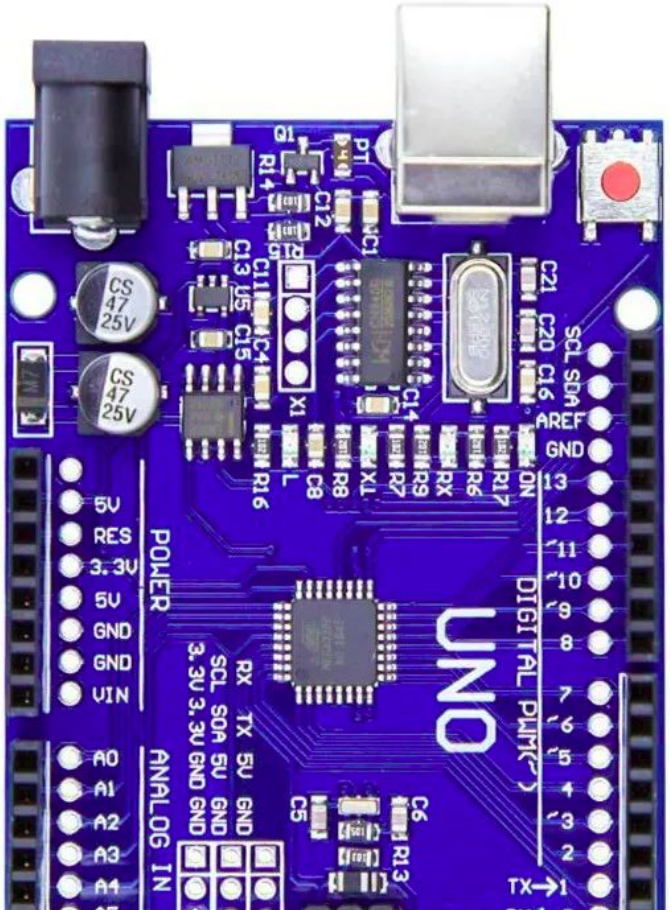
Comunicación con el exterior

Comunicación con el exterior
GPIO (general-purpose input/output)
Puertos de entrada salida de propósito general

- Pines que pueden configurarse como entradas o salidas digitales.
  - Entrada:
    - Se lee como 1 lógico si se aplica un nivel alto de tensión en el pin.
    - Se lee como 0 lógico si se aplica un nivel bajo de tensión en el pin.
  - Salida:
    - Produce un nivel alto de tensión en el pin si se escribe un 1 lógico.
    - Produce un nivel bajo de tensión en el pin si se escribe un 0 lógico.
- Usualmente agrupados en grupos de 8 llamados “puertos”.
- Pueden tener resistencias pull-up o pull-down.
  - Pull-up: imponen un 1 lógico por defecto.
  - Pull-down: imponen un 0 lógico por defecto.

Entrada Salida con el IDE de Arduino UNO

```
/*Configuración de pines como
entrada o salida*/
pinMode(pin, OUTPUT);
pinMode(pin, INPUT);
/*Leer el valor de un pin*/
valor=digitalRead(pin);
/*Escribe val en el pin indicado
value puede valer HIGH o LOW. */
digitalWrite(pin, value);
```



Ejemplo de especificaciones

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V	-0.5		$0.2V_{CC}$(1)	V
		V_{CC} = 2.4V - 5.5V	-0.5		$0.3V_{CC}$(1)	
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V	$0.7V_{CC}$(2)		V_{CC} + 0.5	V
		V_{CC} = 2.4V - 5.5V	$0.6V_{CC}$(2)		V_{CC} + 0.5	

Ejemplo: Si Vcc=5.0V
Un 0 será: de -0.5 V a 1.5V
Un 1 será: de 3.0 V a 5.5V

Valores máximos:

Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA

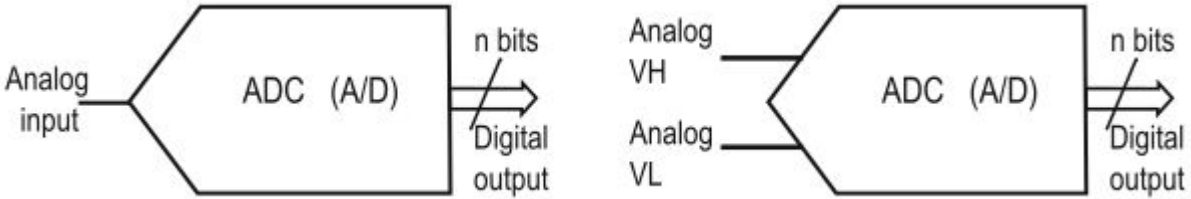
Figura tomada de: Atmel, “8-bit AVR Microcontrollers. ATmega328-328P Datasheet complete”, página 365

Entrada Salida en Raspbian

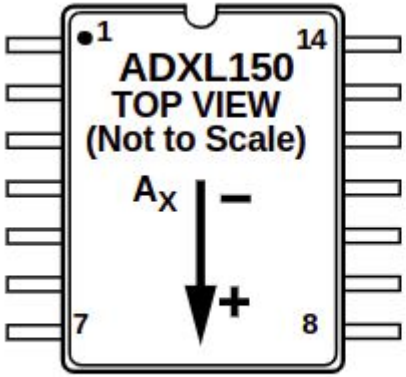
- La entrada/salida se controla mediante la escritura a archivos virtuales.
- Para habilitar un pin como GPIO, debemos escribir el número de pin en el archivo "*/sys/class/gpio/export*".
  - Esto crea la carpeta "/sys/class/gpio/gpioX" siendo X el número de pin.
  - *Escribiendo in o out en "/sys/class/gpio/gpioX/direction" se configura como entrada o salida.*
  - *Leer o escribir el pin "/sys/class/gpio/gpioX/value"*
  - Para deshabilitar el GPIO usado, debe escribirse el número de pin en "/sys/class/gpio/unexport".

Conversor analógico a digital

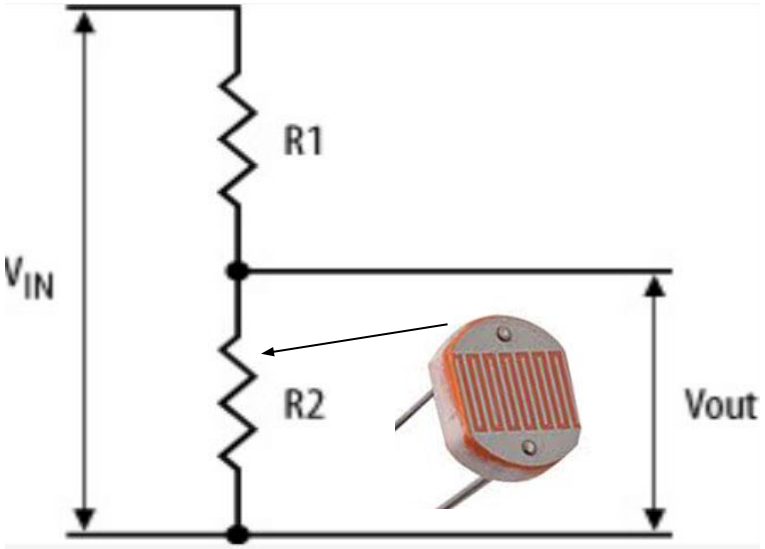
- Conversor Analógico a Digital, ADC o A/D: convierte una tensión (voltios) entre dos pines de analógico a un número digital de n bits.
- Puede tener:
  - Una entrada: mide diferencia de potencial entre la entrada y tierra (0 volts).
  - Dos entradas: mide diferencia de potencial entre esas dos entradas.



Conversor analógico a digital



POSITIVE A = POSITIVE V_{OUT}



$$V_{OUT} = V_{IN} * R2 / (R1 + R2)$$

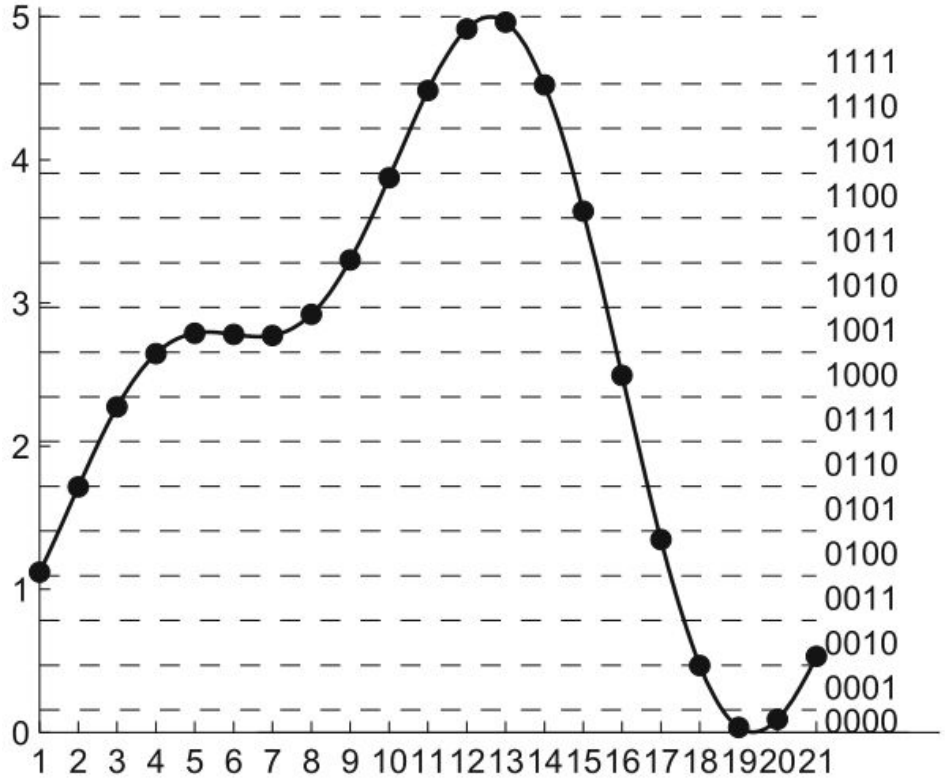
Conversor analógico a digital

Cuantización y Resolución

V_{REF}=5.00V
Resolución: 4 bits
Resolución (LSB) = $V_{REF}/2^4$=0,3125V
(LSB: least significant bit)

V_X=1.0 Salida: 0011
V_X=2.0 Salida: 0110

V_X=1.88 Salida: 0110 (1.88/0,3125=6.016)
V_X=2.18 Salida: 0110 (2.18/0,3125=6.976)
V_X=2.19 Salida: 0111



Conversor analógico a digital

Resoluciones típicas (suponiendo V_{REF}=5.00V):

Resolución bits	Resolución Volts
8 bits	19.5 mV
10 bits	4.88 mV
12 bits	1.22 mV
16 bits	0.076 mV

Más bits, más resolución, mayor tiempo de conversión

Exactitud

Errores producidos por diferentes factores:

- Calidad de fabricación
- Distorsión introducida por los componentes.
- Ruido
- Usualmente especificada como un porcentaje de la resolución.

Ejemplo: Conversor AD Atmega 328

28.1. Features

- 10-bit Resolution
- 0.5 LSB Integral Non-Linearity
- ±2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- Six Multiplexed Single Ended Input Channels
- Two Additional Multiplexed Single Ended Input Channels (TQFP and VFQFN Package only)
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

V_{REF}=5.0V
LSB=5.0V/(2^{10})=0,00488V
Lectura=X±0,00977 V

SPS: samples per second

Figura obtenida de: Atmel, “8-bit AVR Microcontrollers. ATmega328-328P Datasheet complete”. Página 305

Conversor A/D en Arduino

```
analogReference(DEFAULT);
```

- DEFAULT: 5 Volts
- INTERNAL: 1.1 V
- EXTERNAL: Voltaje aplicado al pin AREF
- otras opciones dependiendo del modelo…

```
pinMode(A0, INPUT);
pinMode(A0, INPUT_PULLUP);
```

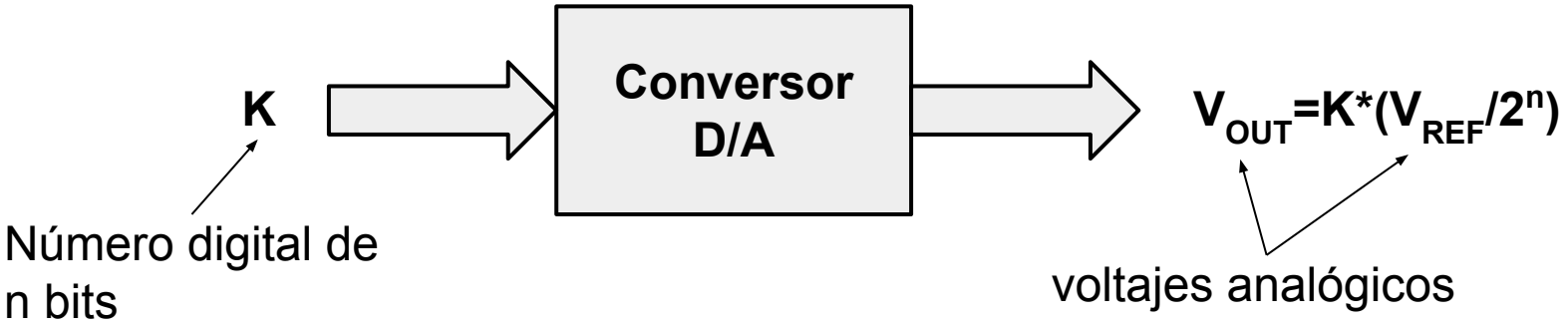
Sin resistencia pullup (sin valor por defecto)
Con resistencia pullup (valor por defecto = V_{REF})

```
analogReadResolution(bits);
```

Solo algunos modelos

```
valor=analogRead(A0);
```

Conversor Digital a analógico



Ejemplo: V_{REF}=5.00V y tenemos 8 bits
Resolución=$V_{REF}/ 2^n$ = 5.00V/(2^8)=0.01953V

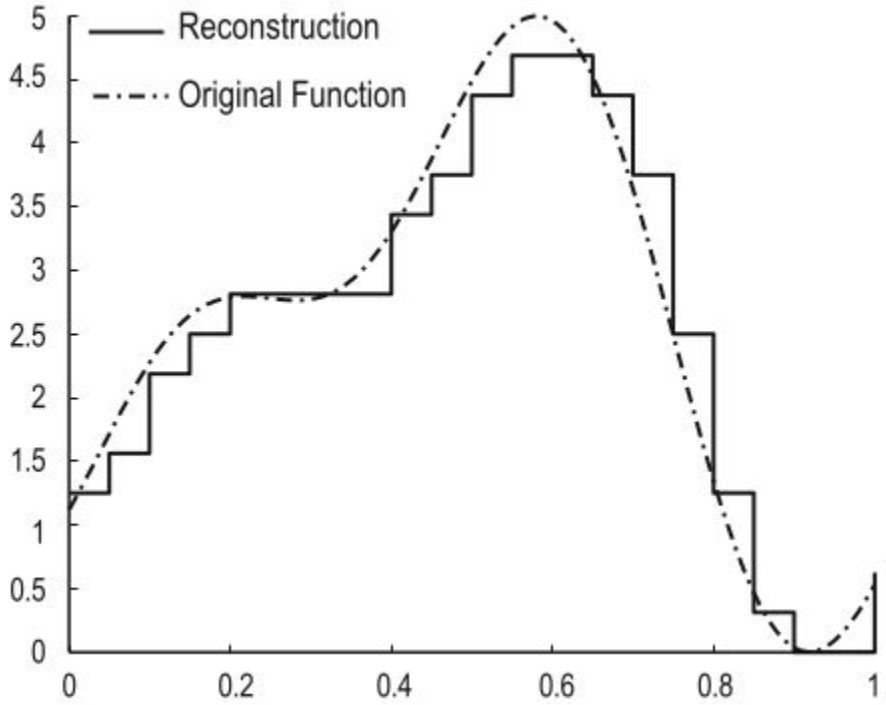
Si K=bx10010010 (146) ; V_{OUT}=2.85V
Si K=bx00110000 (48) ; V_{OUT}=0.937V

V_{REF} puede ser interno o externo

Conversor Digital a analógico

- Control de velocidad de motores
- Control de calefacción
- Control del nivel de iluminación
- Reconstruir una señal (por ejemplo: audio)

Conversor Digital a Analógico - Reconstrucción



Conversor D/A en Arduino

- Conversor tipo PWM (ver filminas siguientes) de 8 bits (256 niveles) o algunos modelos 12 bits (4096 niveles).

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)

```
pinMode(pin, OUTPUT);
```

```
analogWriteResolution(resolución);
```
Algunos modelos permiten elegir entre 8 y 12 bits.

```
analogWrite(pin,valor);
```
valor: entre 0 y 256 o 4096 (algunos modelos)

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENI

Conversor Digital a analógico
Conversor basado en PWM (Pulse Width Modulation)

Duty cycle: fracción del tiempo total que el pulso está en alto.

$V_{MEDIA}=V_{REF}*Duty\_cycle$

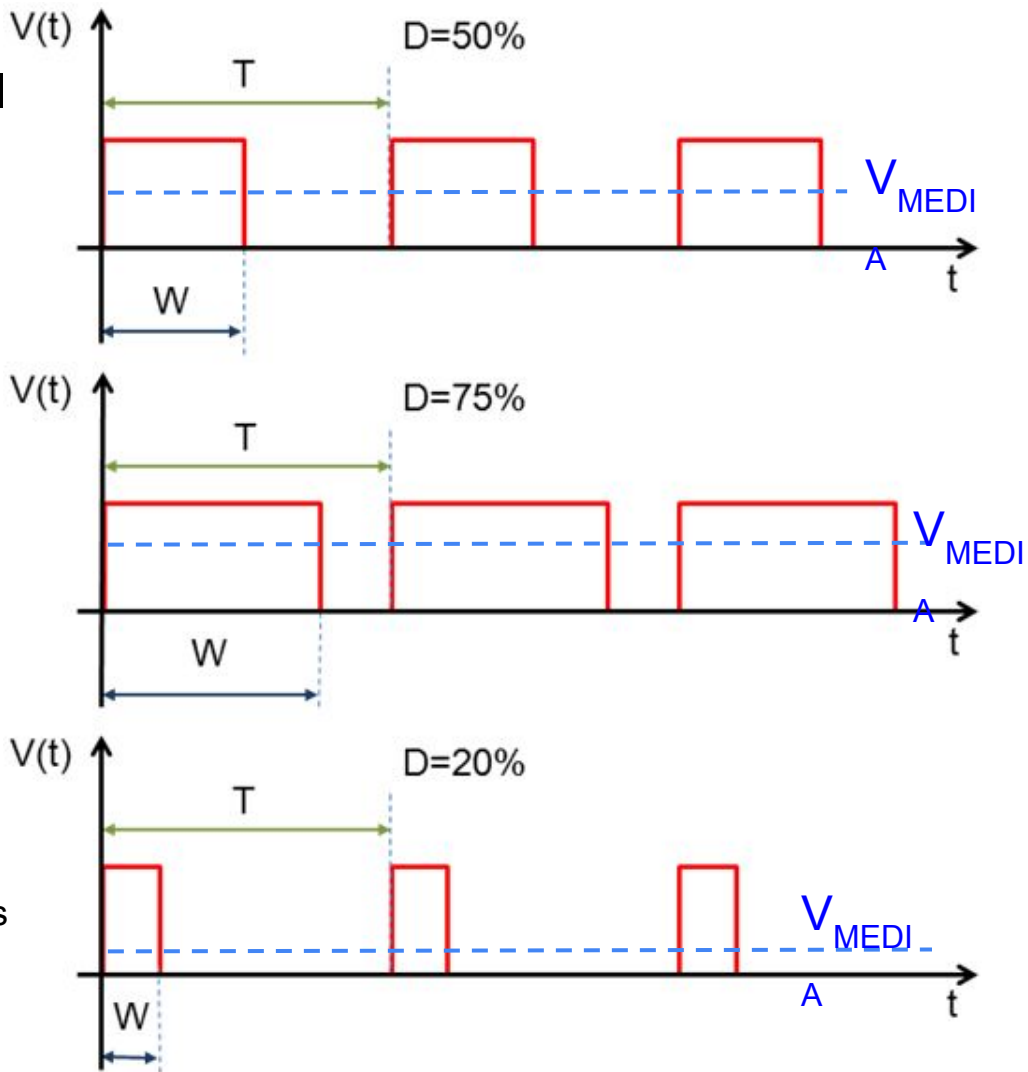
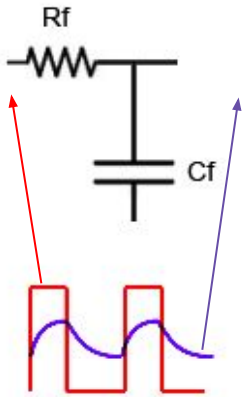
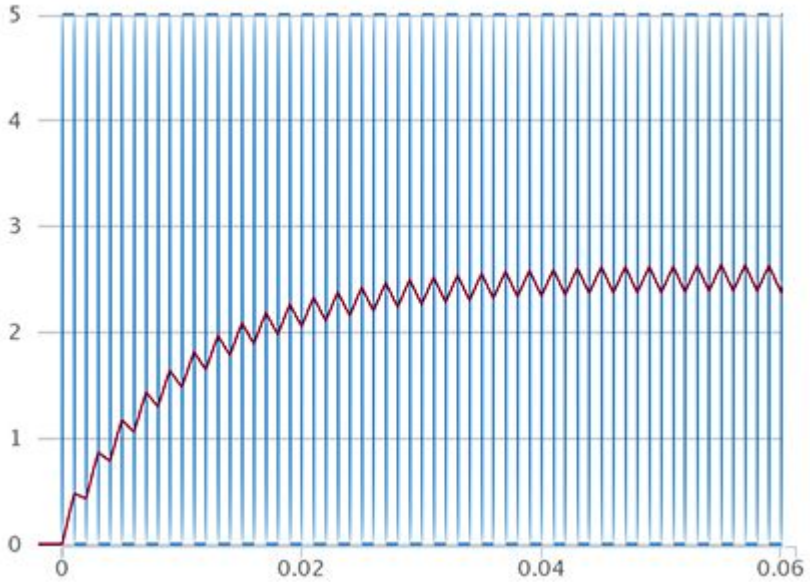


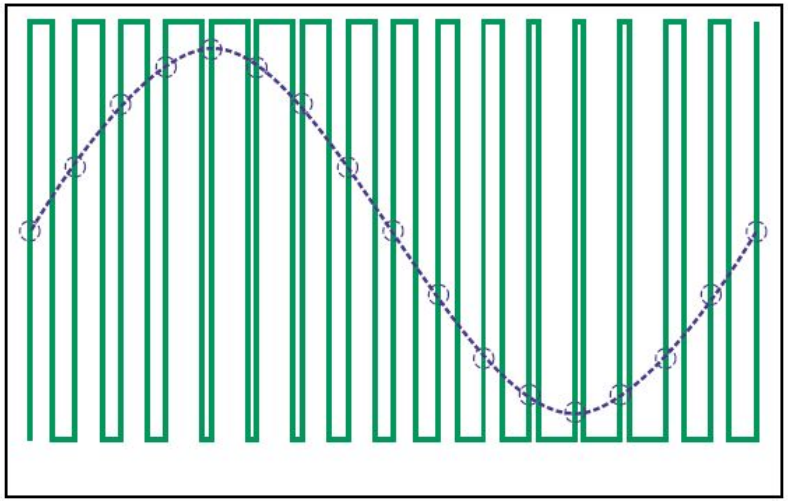
Figura basada en: Chaves Osorio, José & Quintero, Edwin & Cortes, Jimy. (2011). Generación de señales senoidales mediante PWM y filtros activos de segundo orden.

Reconstrucción de una señal analógica a partir de un PWM

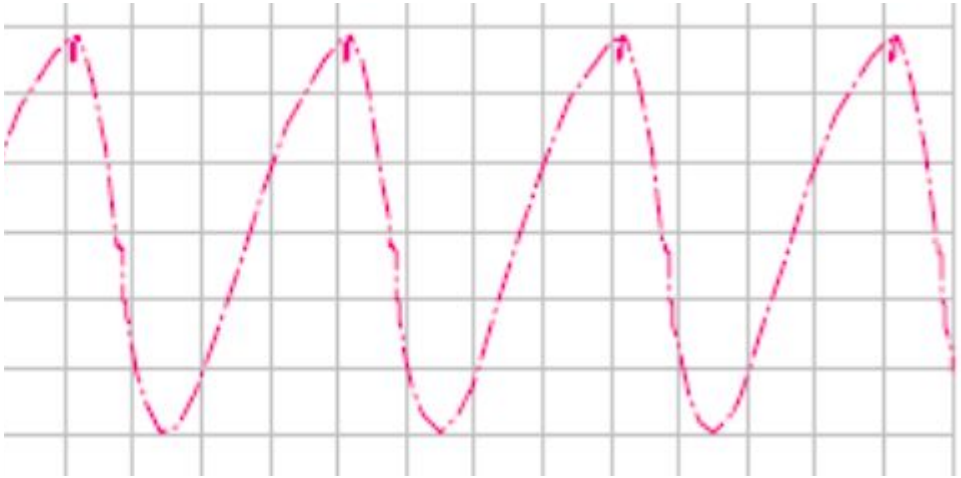




Reconstrucción de una señal analógica a partir de un PWM



PWM y señal senoidal (ideal)



Señal senoidal resultante de un conversor D/A tipo PWM más filtro

Figura basada en: Chaves Osorio, José & Quintero, Edwin & Cortes, Jimy. (2011). Generación de señales senoidales mediante PWM y filtros activos de segundo orden.

Protocolos de comunicación

- Algunos microcontroladores tienen poder de procesamiento suficiente para utilizar IP, TCP y UDP, poseen Ethernet o WiFi, incluso permiten instalar servidores web (por ejemplo: Apache). Ejemplo: Raspberry.
  - **Ventaja: El sistema puede comunicarse directamente con otras computadoras por medios usuales (servicio web, ssh, etc.).**
  - Desventaja: Requiere un microcontrolador con elevado poder de procesamiento, memoria y puertos Ethernet o Wifi.
- Sistemas embebidos de menor poder de procesamiento o sensores específicos: Se requieren protocolos que requieran menores requisitos
  - USB (serial), SPI, I^2C, CAN, RFID y NFC, ZigBee, LoRa.

UART (universal asynchronous receiver/transmitter)

- No hay línea de clock. Con cada bits de start se sincronizan los relojes.
- Emisor y receptor son básicamente registros de desplazamiento.
- Transmisor y receptor deben acordar bps, longitud de palabra (7 u 8 bits), bits de stop y paridad.

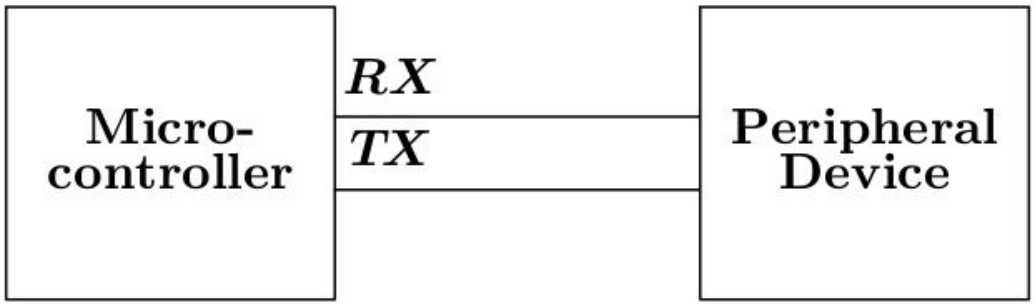


Figura obtenida de: David Russel, "Introduction to Embedded System Using ANSI C and the Arduino Development Environment", página 171.

UART (universal asynchronous receiver/transmitter)

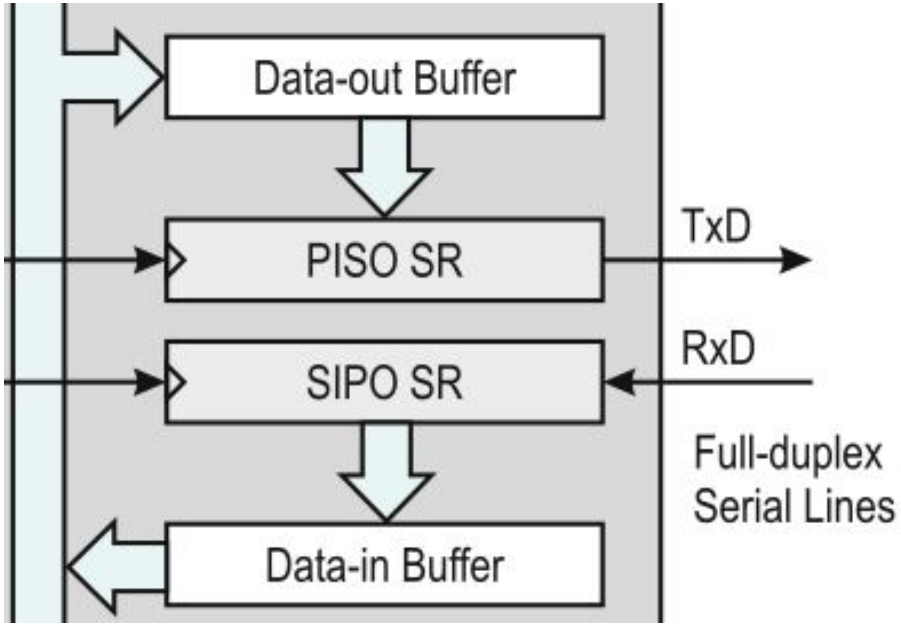


Figura obtenida de: Jiménez et. al. “Introduction to Embedded Systems Using Microcontrollers and the MSP430”, página 483.

Bibliografía:

- John Davies, "MSP430 Microcontroller Basics”. Elsevier. 2008.
- Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier, “Introduction to Embedded Systems Using Microcontrollers and the MSP430”. Springer. 2014.
- Peter Marwedel, "Embedded System Design. Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things". Springer. 2018.
- Perry Xiao ,"Designing Embedded Systems and the Internet of Things (IoT) with the ARM Mbed". Wiley. 2018.
- Microchip, PIC12F629/675 Data Sheet