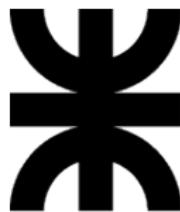


Informática II

Programación del microcontrolador ATmega328

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

El ATmega328

Es un microcontrolador de 8-bits de arquitectura AVR RISC mejorado.
Algunas de las características de este μ C son:

- ▶ Arquitectura RISC avanzada:
 - ▶ 131 instrucciones (la mayoría de las cuales se ejecutan en un ciclo de reloj)
 - ▶ 32 registros de propósitos generales de 8-bits
 - ▶ Máximo de 20MIPS (million instructions per second) @ 20MHz de frecuencia de reloj
 - ▶ Multiplicación por hardware

El ATmega328

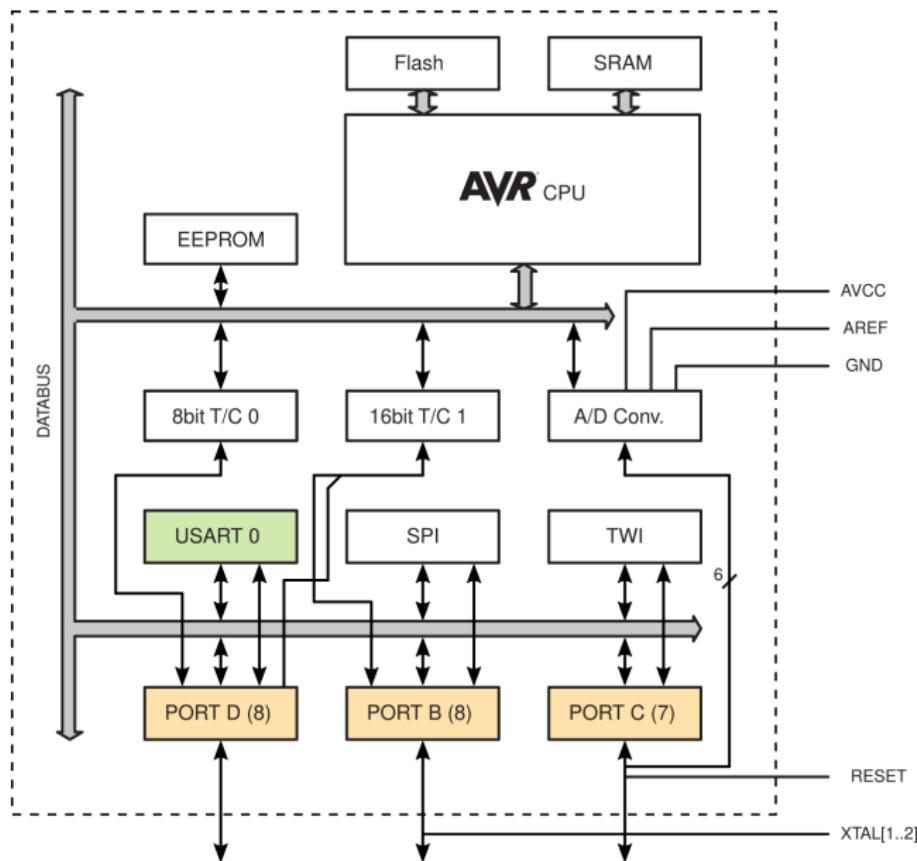
Es un microcontrolador de 8-bits de arquitectura AVR RISC mejorado.
Algunas de las características de este μ C son:

- ▶ Arquitectura RISC avanzada:
 - ▶ 131 instrucciones (la mayoría de las cuales se ejecutan en un ciclo de reloj)
 - ▶ 32 registros de propósitos generales de 8-bits
 - ▶ Máximo de 20MIPS (million instructions per second) @ 20MHz de frecuencia de reloj
 - ▶ Multiplicación por hardware
- ▶ Memoria:
 - ▶ 32KB de memoria FLASH
 - ▶ 2KB de memoria SRAM (Static Random Access Memory)
 - ▶ 1KB de memoria EEPROM (Electrically Erasable Programmable Read-Only Memory)

El ATmega328

- ▶ Periféricos:
 - ▶ 23 entrada-salida de propósito general (GPIO: General Purpose Input-Output)
 - ▶ 6 canales de PWM (Pulse Width Modulation)
 - ▶ 6/8 canales de ADC (Analog to Digital Converter) de 10-bits
 - ▶ Puerto serial programable (USART: Universal Synchronous/Asynchronous Receiver/Transmitter)
 - ▶ Interfaz serial SPI (Serial Peripheral Interface) maestro-esclavo
 - ▶ Interfaz serial de 2-cables (compatible con el I²C de Philips)
 - ▶ Interrupción externa por cambio de nivel en entrada digital

El ATmega328 – Diagrama en bloques

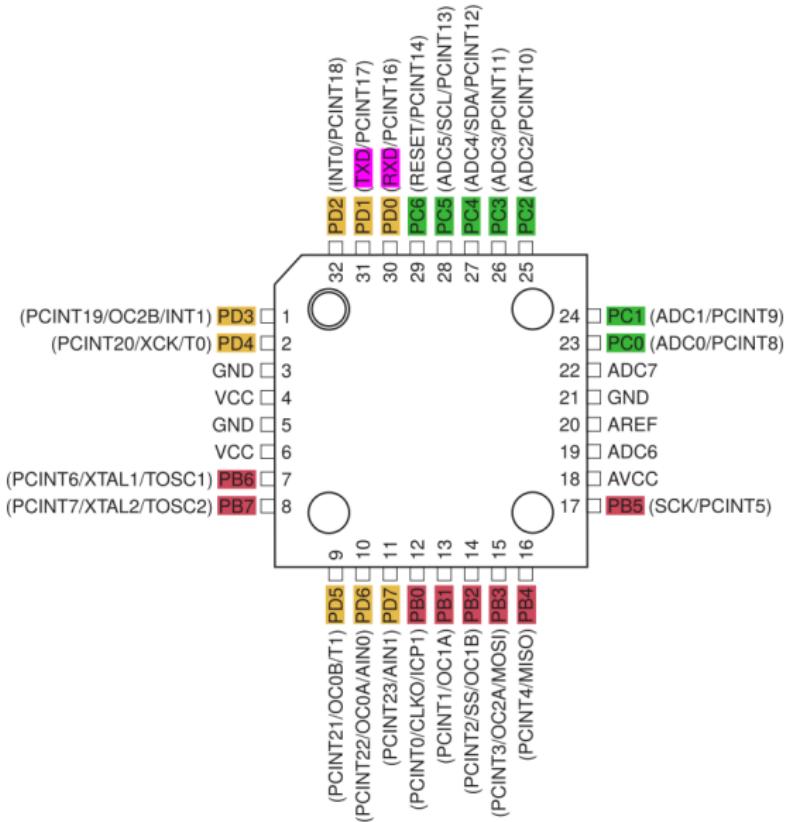


El ATmega328 – Encapsulados

28 PDIP	
(PCINT14/RESET) PC6	1
(PCINT16/ RXD) PD0	2
(PCINT17/ TXD) PD1	3
(PCINT18/INT0) PD2	4
(PCINT19/OC2B/INT1) PD3	5
(PCINT20/XCK/T0) PD4	6
VCC	7
GND	8
(PCINT6/XTAL1/TOSC1) PB6	9
(PCINT7/XTAL2/TOSC2) PB7	10
(PCINT21/OC0B/T1) PD5	11
(PCINT22/OC0A/AIN0) PD6	12
(PCINT23/AIN1) PD7	13
(PCINT0/CLKO/ICP1) PB0	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
PC5 (ADC5/SCL/PCINT13)	
PC4 (ADC4/SDA/PCINT12)	
PC3 (ADC3/PCINT11)	
PC2 (ADC2/PCINT10)	
PC1 (ADC1/PCINT9)	
PC0 (ADC0/PCINT8)	
GND	
AREF	
AVCC	
PB5 (SCK/PCINT5)	
PB4 (MISO/PCINT4)	
PB3 (MOSI/OC2A/PCINT3)	
PB2 (SS/OC1B/PCINT2)	
PB1 (OC1A/PCINT1)	

El ATmega328 – Encapsulados

32 TQFP Top View



El ATmega328 – Memoria

Tiene dos espacios de memorias diferentes (arquitectura Harvard):

1. una es la memoria de programa y
2. otra la memoria de datos.

El ATmega328 – Memoria

Tiene dos espacios de memorias diferentes (arquitectura Harvard):

1. una es la memoria de programa y
2. otra la memoria de datos.

Memoria de programa (Flash)

- ▶ La memoria Flash, para almacenar el programa, es de $16K \times 16\text{-bits}$ haciendo un total de 32KBytes ($32K \times 8\text{-bits}$).
- ▶ El contador de programa (PC: Program Counter) es de 14-bits lo que permite direccionar 16K ubicaciones de la memoria de programa.

El ATmega328 – Memoria

Tiene dos espacios de memorias diferentes (arquitectura Harvard):

1. una es la memoria de programa y
2. otra la memoria de datos.

Memoria de programa (Flash)

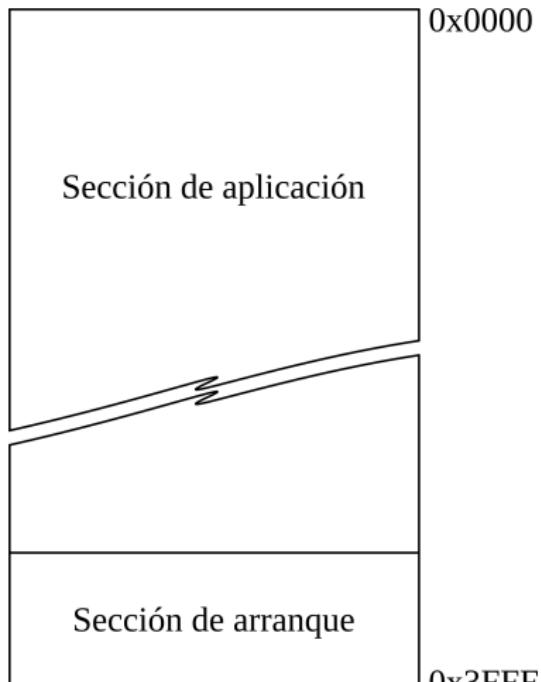
- ▶ La memoria Flash, para almacenar el programa, es de $16K \times 16\text{-bits}$ haciendo un total de 32KBytes ($32K \times 8\text{-bits}$).
- ▶ El contador de programa (PC: Program Counter) es de 14-bits lo que permite direccionar 16K ubicaciones de la memoria de programa.

Memoria de programa (RAM)

La memoria RAM se organiza en diferentes secciones.

- ▶ Las primeras direcciones para acceder a los registros de propósito general y los de entrada-salida,
- ▶ las siguientes para almacenar las variables del programa.

El ATmega328 – Memoria

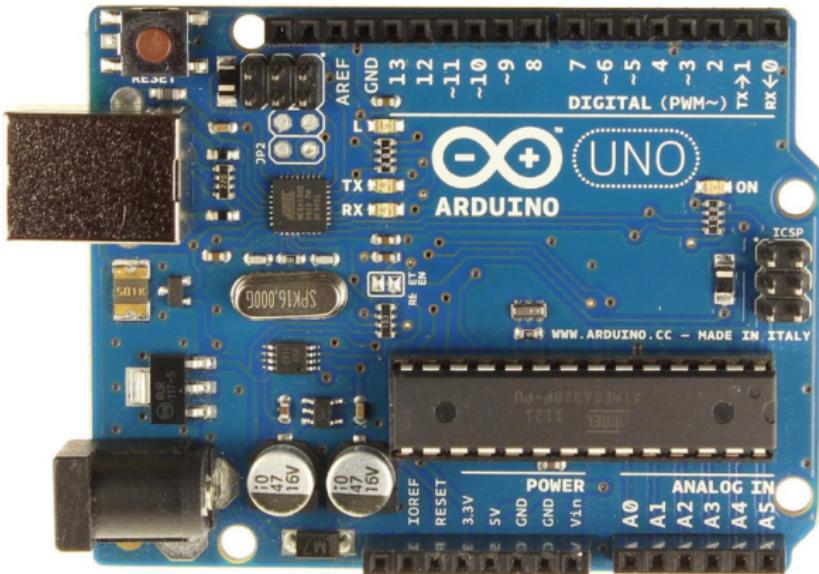


Mapa de mem. Flash

Mapa de mem. RAM

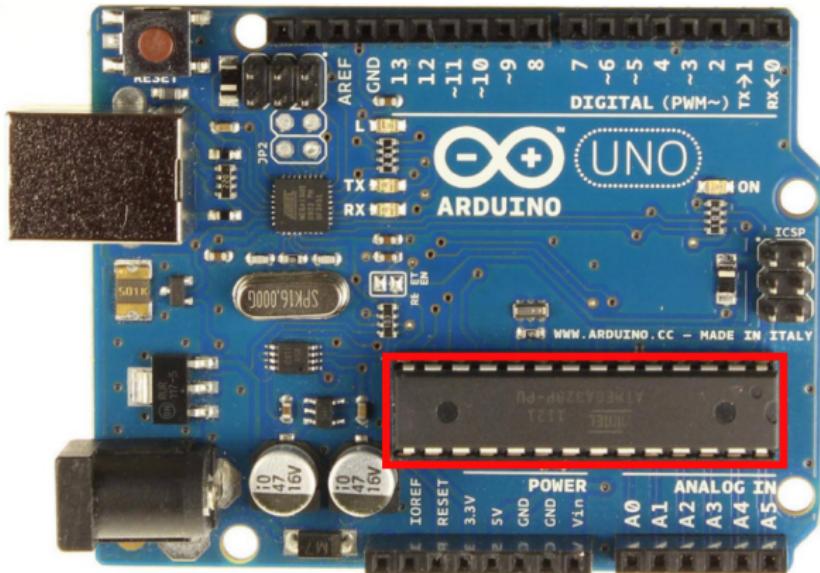
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



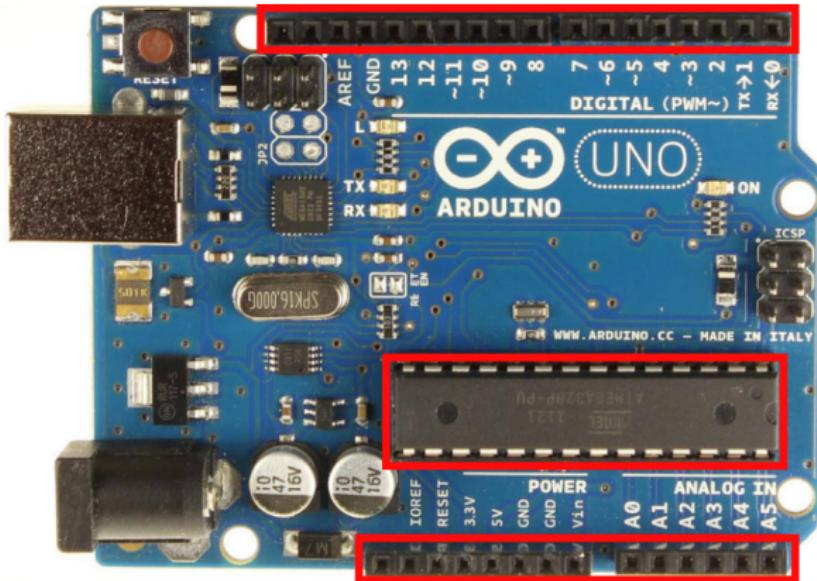
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



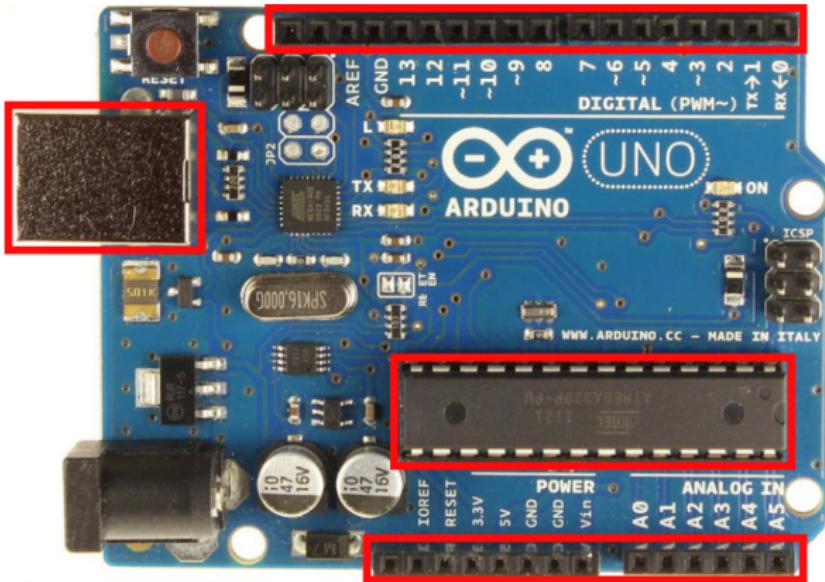
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



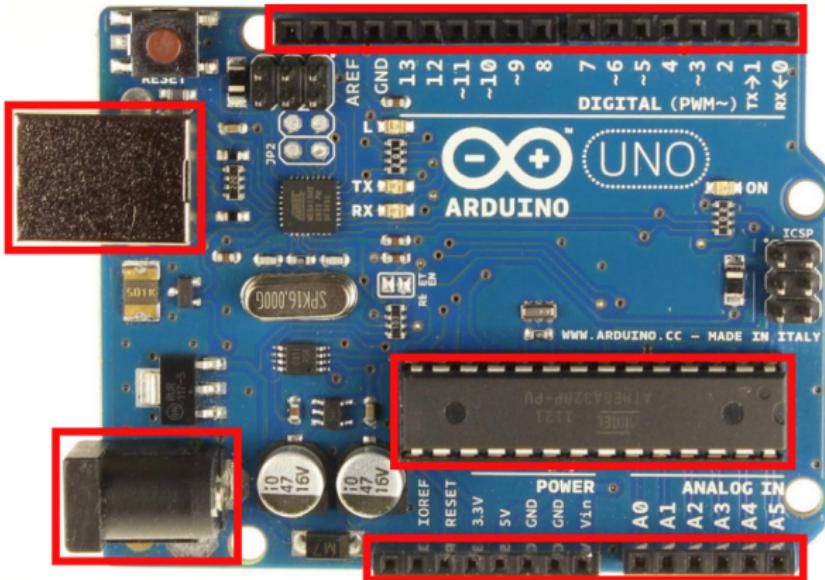
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



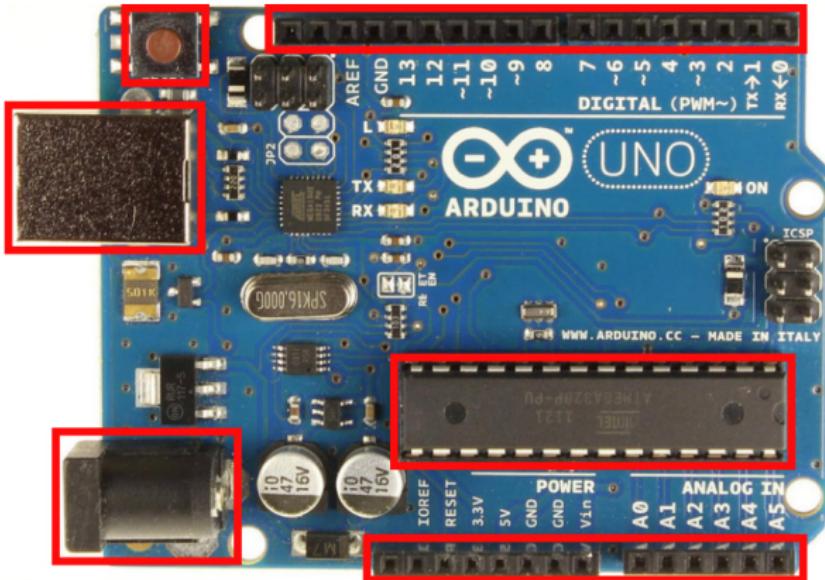
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



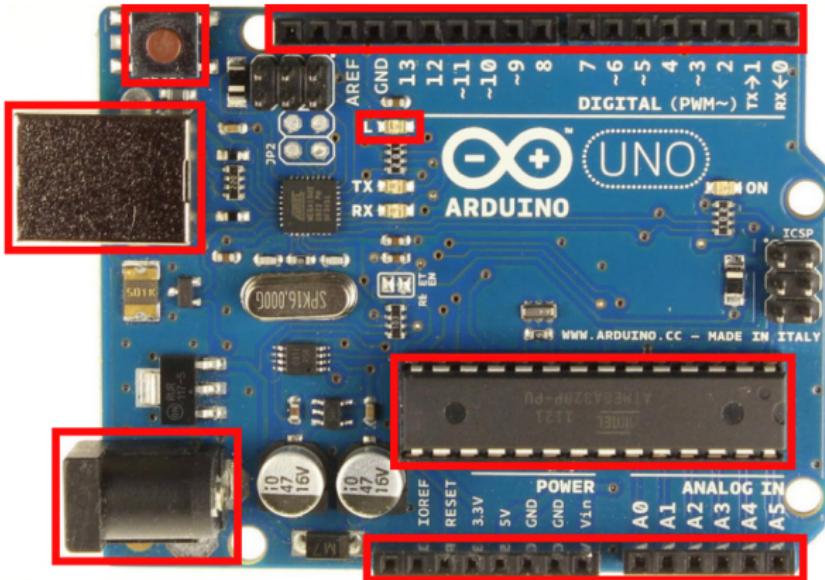
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



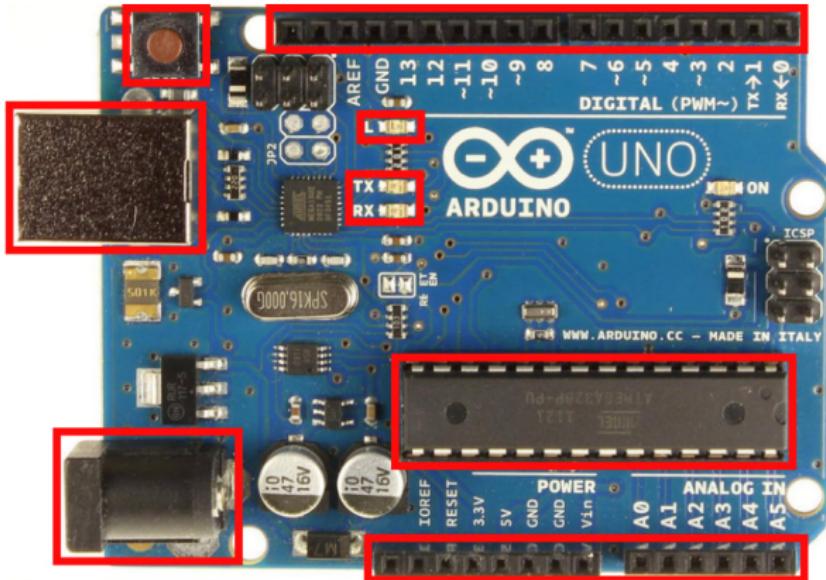
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



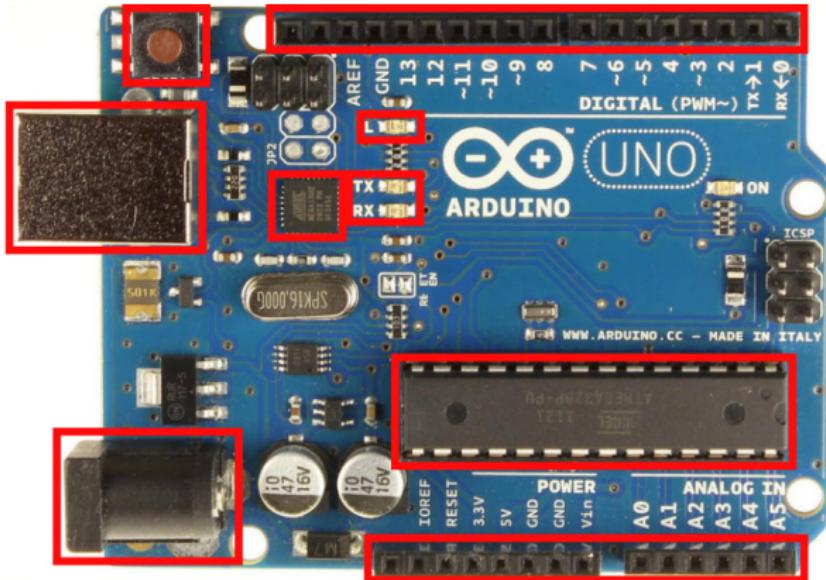
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



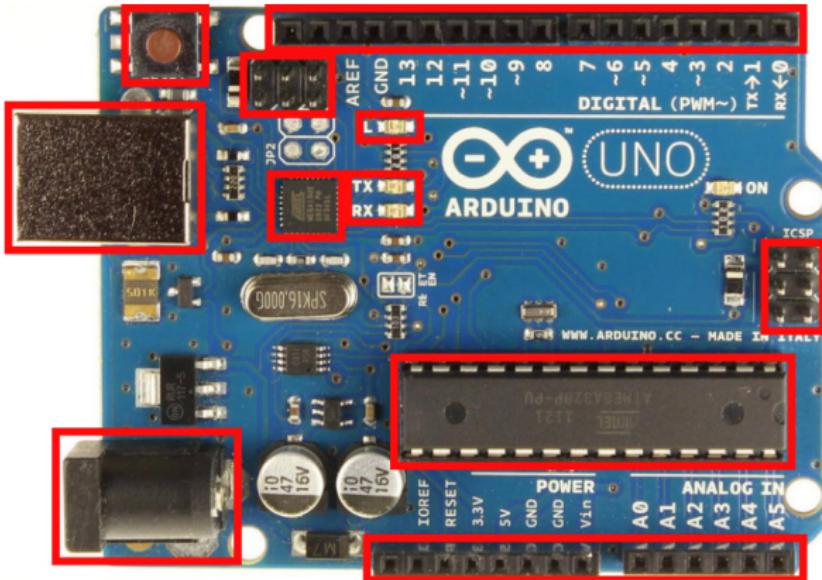
Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.



Arduino UNO Rev3

La placa Arduino UNO tiene un microcontrolador (μ C) marca Microchip® (anteriormente Atmel®) modelo ATmega328.

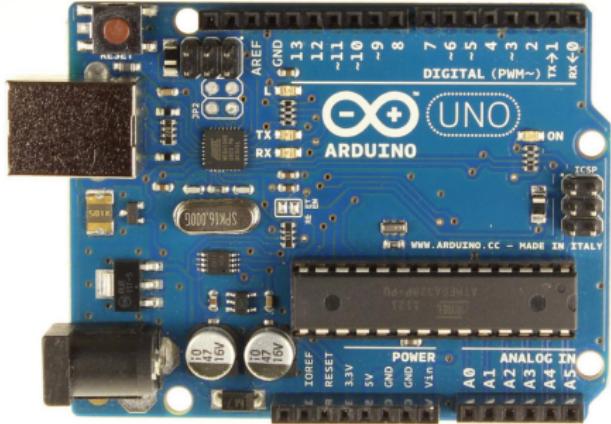


Arduino UNO Rev3 – Algunas características

Algunas de las características de esta placa son:

- ▶ Tensión de funcionamiento de 5V
- ▶ Tensión de entrada de 7 a 12V (de 6 a 20V como rango máximo)
- ▶ 14 pines digitales de entrada/salida
- ▶ 6 pines digitales con función PWM
- ▶ 6 pines de entrada analógica

Arduino UNO – Versiones



Arduino UNO Rev3:

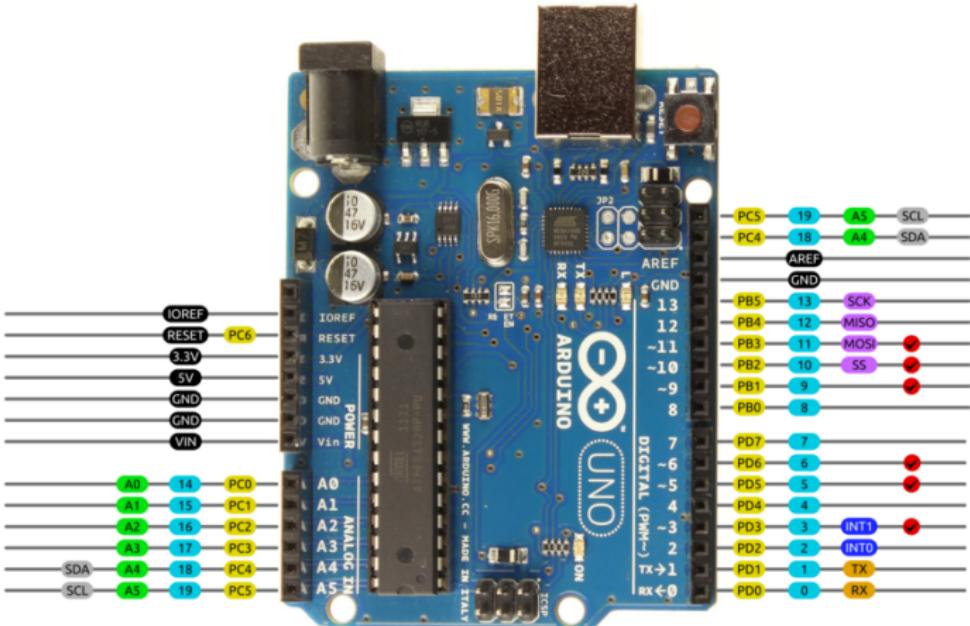
- ▶ ATmega328P DIP (Dual In-line Package)
- ▶ ATmega16U2 μ C con controlador USB, QFN (Quad-flat No-leads)



Arduino UNO CH340:

- ▶ ATmega328P TQFP (Thin Quad Flat Package)
- ▶ CH340, Chip USB a serial

Arduino UNO – Periféricos

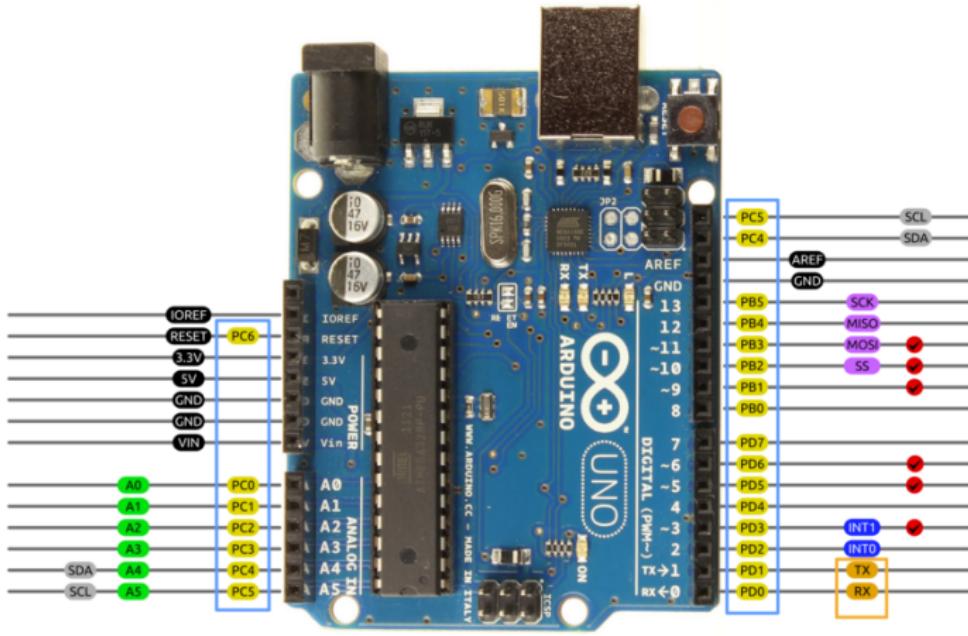


AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT



2014 by Bouni
Photo by Arduino.cc

Arduino UNO – Periféricos



Programación con IDE Arduino



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer

Windows ZIP file

Windows app Win 8.1 or 10



Linux 32 bits

Linux 64 bits

Linux ARM 32 bits

Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#)

[Checksums \(sha512\)](#)

- ▶ Descargar y descomprimir el archivo (`arduino-1.8.19-linux64.tar.xz`)
- ▶ Ejecutar el script de instalación (`./install.sh`)

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.

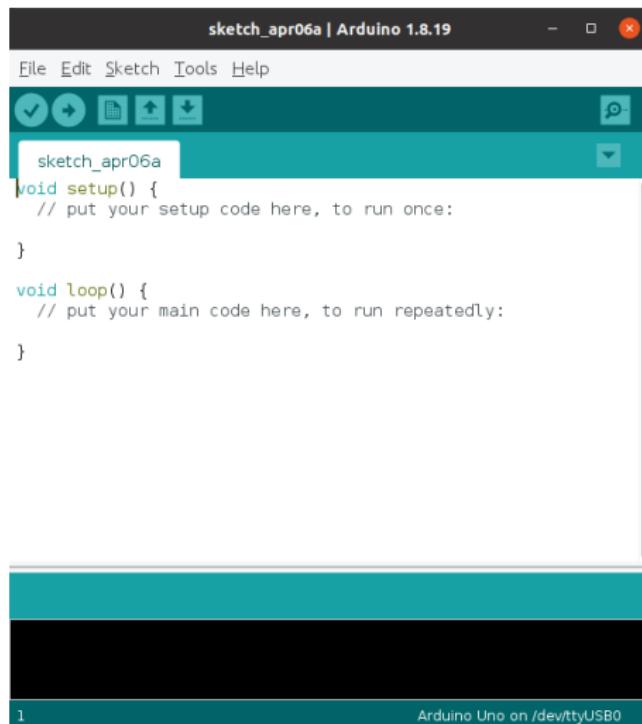
The screenshot shows the Arduino IDE interface. The title bar reads "sketch_apr06a | Arduino 1.8.19". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for save, upload, and search. A dropdown menu shows "sketch_apr06a". The main code editor contains the following code:

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

The status bar at the bottom indicates "Arduino Uno on /dev/ttyUSB0".

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.

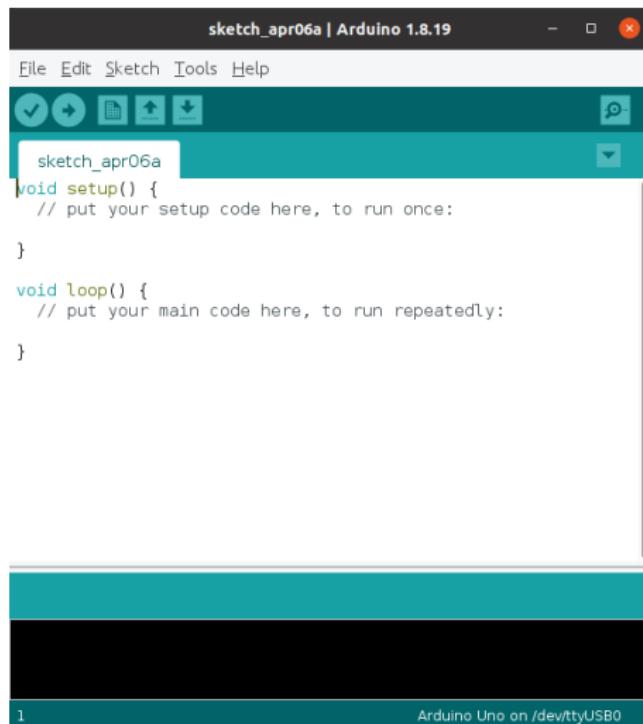


Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

sketch Arduino – 2 bloques:

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



The screenshot shows the Arduino IDE interface. The title bar reads "sketch_apr06a | Arduino 1.8.19". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for save, upload, and search. The main area displays the code for "sketch_apr06a":

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

The status bar at the bottom indicates "Arduino Uno on /dev/ttyUSB0".

Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

sketch Arduino – 2 bloques:

- ▶ **setup()**: se ejecuta una única vez cuando se enciende o resetea la placa

Programación con IDE Arduino

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

sketch Arduino – 2 bloques:

- ▶ **setup()**: se ejecuta una única vez cuando se enciende o resetea la placa
- ▶ **loop()**: se ejecuta de forma constante (bucle)

Sketch ejemplo: blink

File->Examples->01.Basics->Blink

```
1 // the setup function runs once when you press reset or
2 // power the board
3 void setup() {
4     // initialize digital pin LED_BUILTIN as an output.
5     pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 // the loop function runs over and over again forever
9 void loop() {
10    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on
11    delay(1000);                      // wait for a second
12    digitalWrite(LED_BUILTIN, LOW);     // turn the LED off
13    delay(1000);                      // wait for a second
14 }
```

Sketch ejemplo: puerto serie

```
1 #define MENSAJE "Hola mundo"
2
3 // La función 'setup' se ejecuta una única vez al
4 // presionar reset o encender la placa
5 void setup() {
6     // Inicializa el puerto serie (UART) a 9600 bps.
7     Serial.begin(9600);
8 }
9
10 // La función 'loop' corre indefinidamente una y otra vez
11 void loop() {
12     // Imprime (envía) la cadena MENSAJE por puerto serie.
13     Serial.println(MENSAJE);
14     delay(500);
15 }
```

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

`binutils`: colección de herramientas para la manipulación de archivos binarios

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

`binutils`: colección de herramientas para la manipulación de archivos binarios

`libc-avr`: subconjunto de la biblioteca estándar de C con funciones específicas para arquitectura AVR

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

`binutils`: colección de herramientas para la manipulación de archivos binarios

`libc-avr`: subconjunto de la biblioteca estándar de C con funciones específicas para arquitectura AVR

`avrdude`: programa para escribir y leer la memoria Flash del μ C

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

`binutils`: colección de herramientas para la manipulación de archivos binarios

`libc-avr`: subconjunto de la biblioteca estándar de C con funciones específicas para arquitectura AVR

`avrdude`: programa para escribir y leer la memoria Flash del μ C

La instalación del Toolchain en Ubuntu se realiza instalando los siguientes paquetes: `gcc-avr`, `binutils-avr`, `avr-libc` y `avrdude`.

Programación con Toolchain GCC para AVR

El Toolchain de GCC para arquitectura AVR está compuesto por:

`gcc`: el compilador de C y C++

`binutils`: colección de herramientas para la manipulación de archivos binarios

`libc-avr`: subconjunto de la biblioteca estándar de C con funciones específicas para arquitectura AVR

`avrdude`: programa para escribir y leer la memoria Flash del μ C

La instalación del Toolchain en Ubuntu se realiza instalando los siguientes paquetes: `gcc-avr`, `binutils-avr`, `avr-libc` y `avrdude`.

- ▶ El compilador GCC para AVR (`avr-gcc`) es en realidad es un compilador cruzado (cross-compiler).
- ▶ Un compilador cruzado es aquel que genera código máquina para una arquitectura diferente a aquella en la cual se ejecuta el compilador.

Programación con Toolchain GCC para AVR

Archivo 'blink.c'

```
1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 #define BLINK_DELAY_MS 1000
5
6 int main (void)
7 {
8     /* Inicializa el pin digital como salida */
9     DDRB |= _BV(DDB5);
10
11    while(1)
12    {
13        PORTB |= _BV(PORTB5); /* Enciende LED */
14        _delay_ms(BLINK_DELAY_MS);
15
16        PORTB &= ~_BV(PORTB5); /* Apaga LED */
17        _delay_ms(BLINK_DELAY_MS);
18    }
19    return 0;
20 }
```

Programación con Toolchain GCC para AVR

Construcción del programa y grabación en la memoria del μ C ATmega328 de la placa Arduino UNO:

Programación con Toolchain GCC para AVR

Construcción del programa y grabación en la memoria del μ C ATmega328 de la placa Arduino UNO:

1. Construcción del código fuente (.c) hasta la etapa de ensamblado, de la cual se obtiene un archivo objeto (.o):

```
avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o  
blink.o blink.c
```

Programación con Toolchain GCC para AVR

Construcción del programa y grabación en la memoria del μ C ATmega328 de la placa Arduino UNO:

1. Construcción del código fuente (.c) hasta la etapa de ensamblado, de la cual se obtiene un archivo objeto (.o):

```
avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o  
blink.o blink.c
```

2. Enlazado con bibliotecas, del cual se obtiene un archivo binario (.elf):
`avr-gcc -mmcu=atmega328p blink.o -o blink.elf`

Programación con Toolchain GCC para AVR

Construcción del programa y grabación en la memoria del μ C ATmega328 de la placa Arduino UNO:

1. Construcción del código fuente (.c) hasta la etapa de ensamblado, de la cual se obtiene un archivo objeto (.o):

```
avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o  
blink.o blink.c
```

2. Enlazado con bibliotecas, del cual se obtiene un archivo binario (.elf):
`avr-gcc -mmcu=atmega328p blink.o -o blink.elf`

3. Conversión del código binario (.elf) a formato objeto hexadecimal Intel (.hex)

```
avr-objcopy -O ihex -R .eeprom blink.elf blink.hex
```

Programación con Toolchain GCC para AVR

Construcción del programa y grabación en la memoria del μ C ATmega328 de la placa Arduino UNO:

1. Construcción del código fuente (.c) hasta la etapa de ensamblado, de la cual se obtiene un archivo objeto (.o):

```
avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o  
blink.o blink.c
```

2. Enlazado con bibliotecas, del cual se obtiene un archivo binario (.elf):
`avr-gcc -mmcu=atmega328p blink.o -o blink.elf`

3. Conversión del código binario (.elf) a formato objeto hexadecimal Intel (.hex)
`avr-objcopy -O ihex -R .eeprom blink.elf blink.hex`

4. Grabación en la memoria de programa del μ C

```
avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0  
-b 115200 -U flash:w:blink.hex
```

