

# Workshop

# Arduino

# Bàsics

Jorge Pérez



@akirasan

# Agenda:



Horario: 19:00 a 20:30



Martes 7 noviembre  
Martes 14 noviembre  
Martes 21 noviembre  
Martes 28 noviembre



**Martes 7:**

INTRODUCCIÓN  
RETO 0. [REMOVED]

**Martes 14:**

INTRODUCCIÓN RETO 1  
RETO 1. [REMOVED]

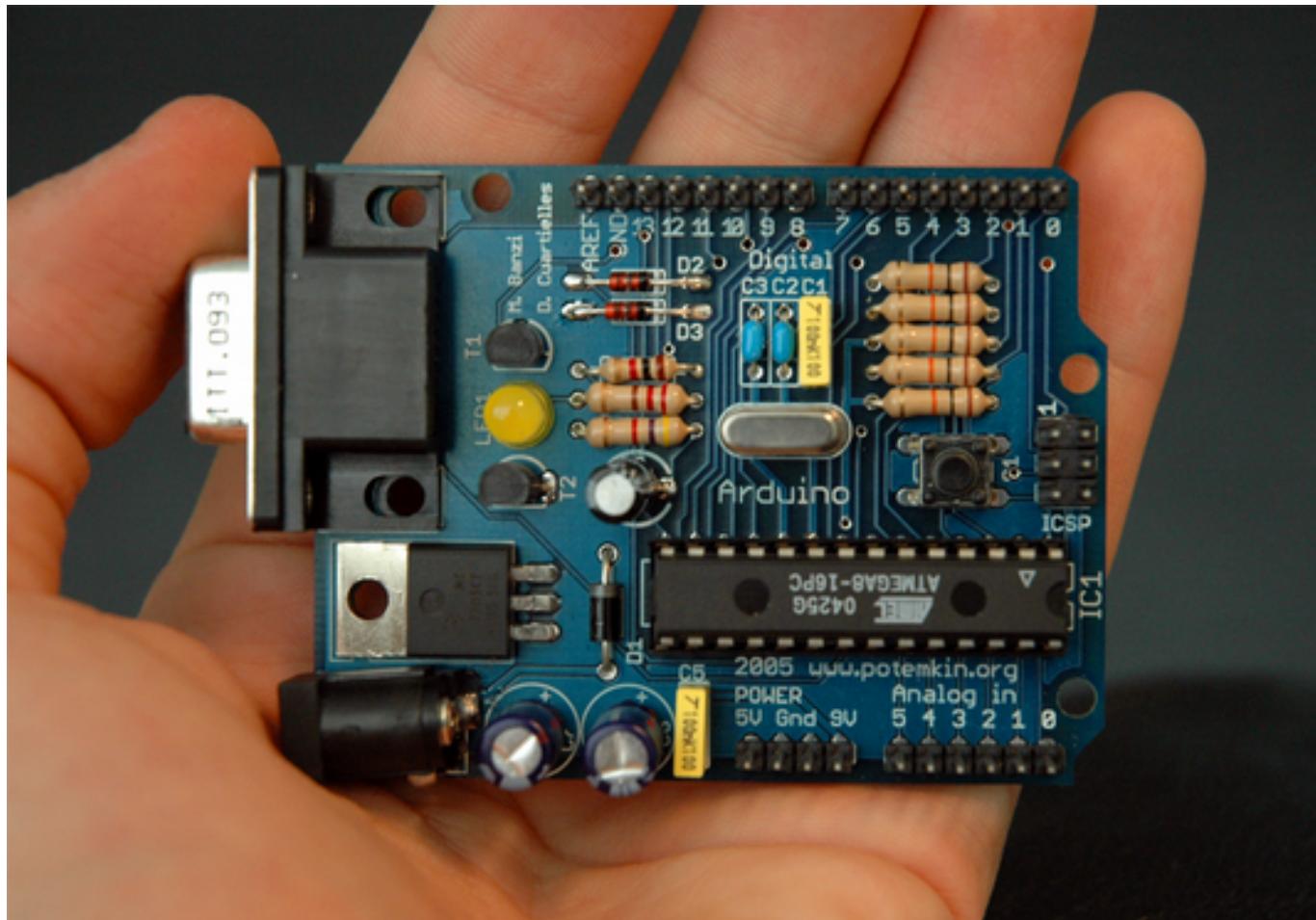
**Martes 21:**

FIN RETO 1  
INTRODUCCIÓN RETO 2. [REMOVED]

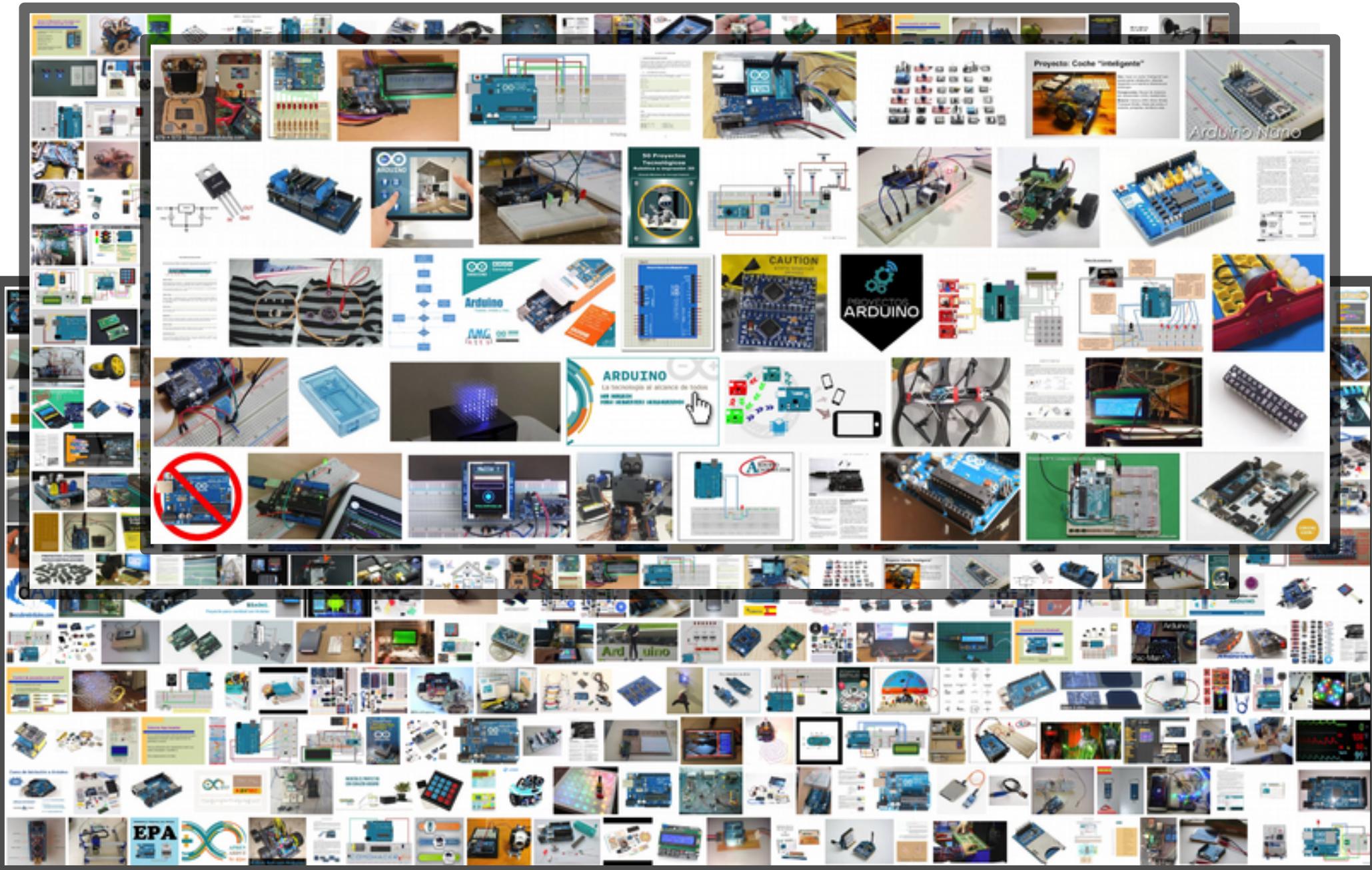
**Martes 28:**

FIN RETO 2

# ¿Qué es Arduino?: Hardware + Software



# ¿Qué es Arduino? : Google



# LOS ORIGENES: Los padres de la criatura



# LOS ORIGENES: Evolución constante



## HISTORY OF ARDUINO

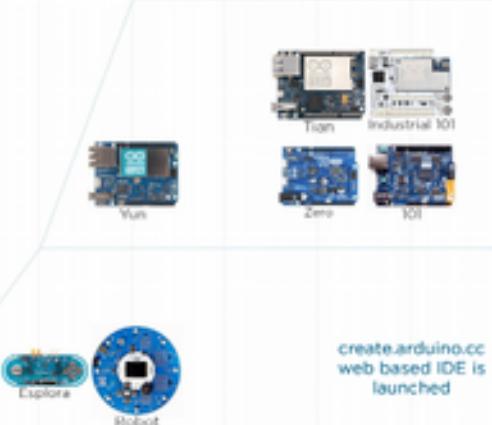
2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

Arduino was born out of the need for a low-cost microcontroller platform for Massimo Banzi's students at the **Interaction Design Institute Ivrea**.

It's named after a local pub: **Bar di Re Arduino**.

The Arduino IDE (Integrated Development Environment) is built upon **Wiring** - a software project written by one of Banzi's students (**Hernando Barragán**). It provides easy-to-use libraries which hide some of the raw C++ going on behind the scenes.

Adafruit estimate 300,000 official boards product



IDE revision 0001 released

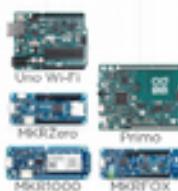
Atmega168 doubles the flash memory

Atmega8 used for the first boards

Atmega328 again doubles the memory



First ever Arduino day  
29/03/14



Arduino Begins



First 32-bit Arduino

Arduino splits:  
arduino.cc (Genuino outside USA) and arduino.org

IDE 1.8 released

Arduino reunites under Arduino Foundation

## ARDUINO TODAY

### Industrial

- Yun/Yun Mini
- Zero
- M0/M0 Pro
- Tian
- 101/Industrial 101



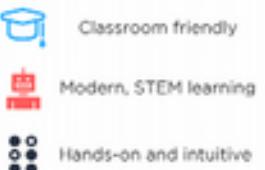
Powerful, smart technology

Rapid prototyping

Easily integrated with other devices

### Educational

- Explora
- Robot



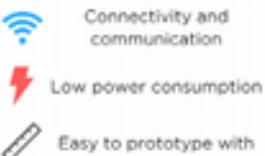
Classroom friendly

Modern, STEM learning

Hands-on and intuitive

### IoT

- MKR1000
- MKRZero
- MKRFOX1200
- Uno Wi-Fi
- Ethernet
- Primo



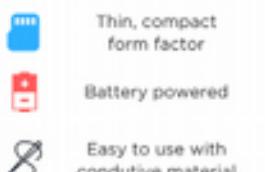
Connectivity and communication

Low power consumption

Easy to prototype with

### Wearables

- LilyPad
- LilyPad Simple
- LilyPad Snap
- LilyPad USB
- Primo Core



Thin, compact form factor

Battery powered

Easy to use with conductive material

### Maker

- Uno
- Leonardo
- Mini/Pro Mini
- Nano/Micro
- Mega 2560/ADK
- Primo
- Due



Affordable

Community driven

Modular and adaptable

# La Familia: Sabores para todos

ENTRY LEVEL	UNO   LEONARDO   101   ROBOT   ESPLORA   MICRO   NANO   MINI  MKR2UNO ADAPTER   STARTER KIT   BASIC KIT   LCD SCREEN
ENHANCED FEATURES	MEGA   ZERO   DUE   MEGA ADK   PRO   MO   MO PRO   MKR ZERO   PRO MINI  MOTOR SHIELD   USB HOST SHIELD   PROTO SHIELD   MKR PROTO SHIELD   4 RELAYS SHIELD  MEGA PROTO SHIELD   MKR RELAY PROTO SHIELD   ISP   USB2SERIAL MICRO  USB2SERIAL CONVERTER
INTERNET OF THINGS	YUN   ETHERNET   TIAN   INDUSTRIAL 101   LEONARDO ETH   MKR FOX 1200   MKR1000  YUN MINI   WIFI SHIELD   WIFI 101 SHIELD   YUN SHIELD   WIRELESS SD SHIELD  WIRELESS PROTO SHIELD   ETHERNET SHIELD V2   GSM SHIELD V2   MKR IoT BUNDLE
EDUCATION	CTC 101
WEARABLE	GEMMA   LILYPAD ARDUINO USB   LILYPAD ARDUINO MAIN BOARD   LILYPAD ARDUINO SIMPLE  LILYPAD ARDUINO SIMPLE SNAP
3D PRINTING	MATERIA 101

BOARDS

MODULES

SHIELDS

KITS

ACCESSORIES

COMING NEXT

# La Familia: Sabores para todos



# LOS ORIGENES: Arduino The Documentary

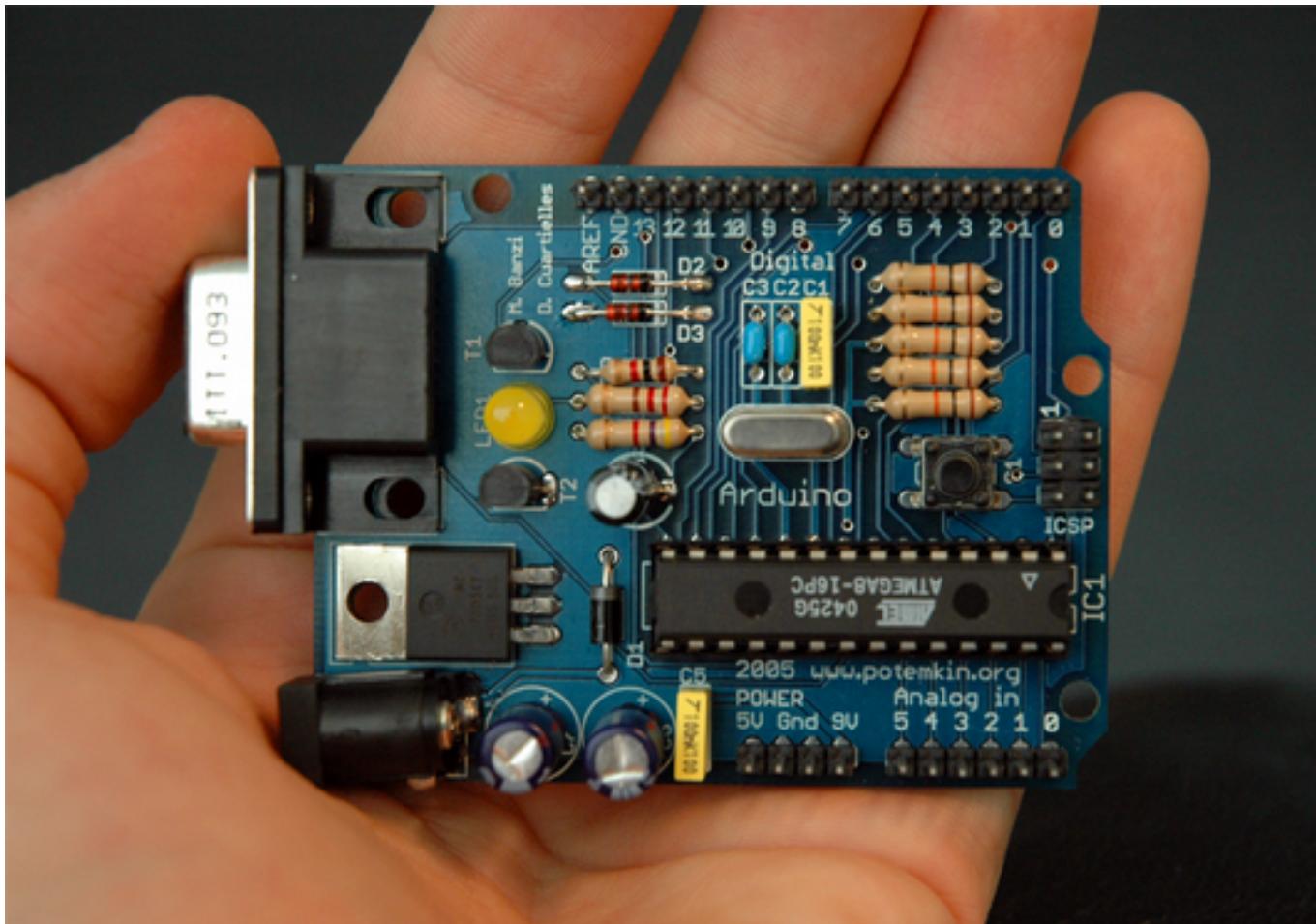


Arduino The Documentary (2010) Spanish HD

Más de Documentales

Reproducir de forma automática el siguiente video

# Hardware/Software: vayamos por partes

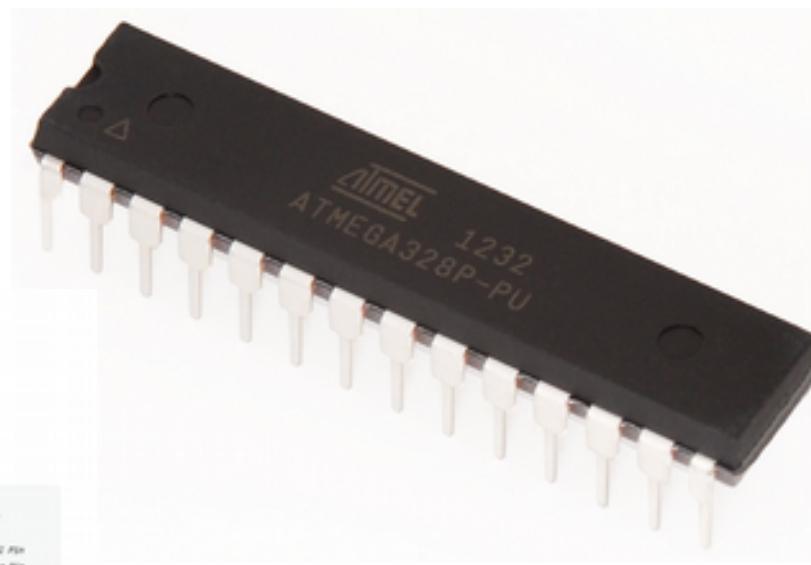
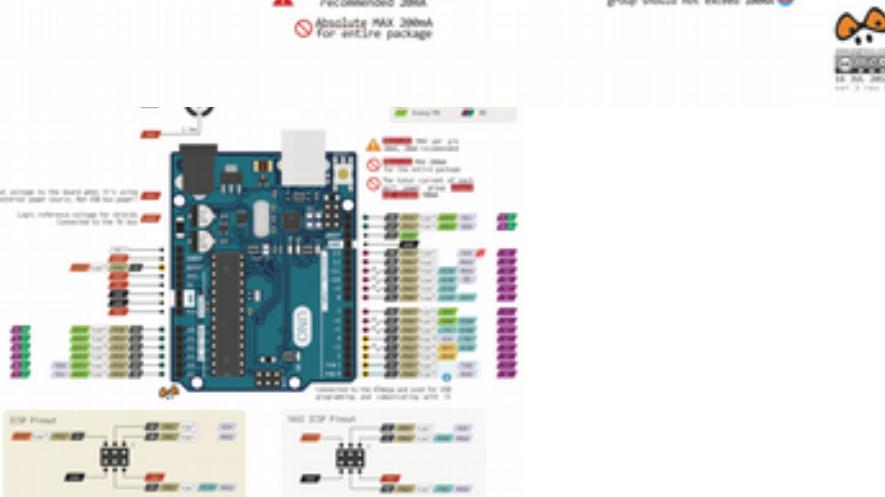
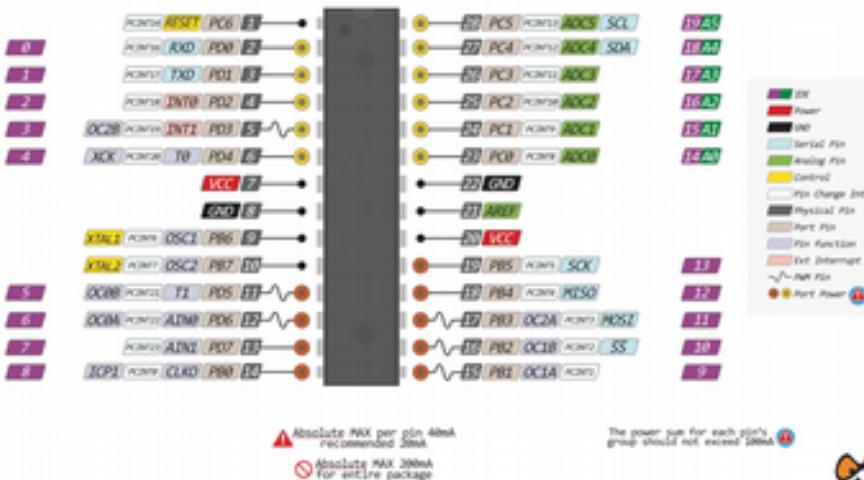


# Hardware

# Hardware: El microprocesador y algo mas...

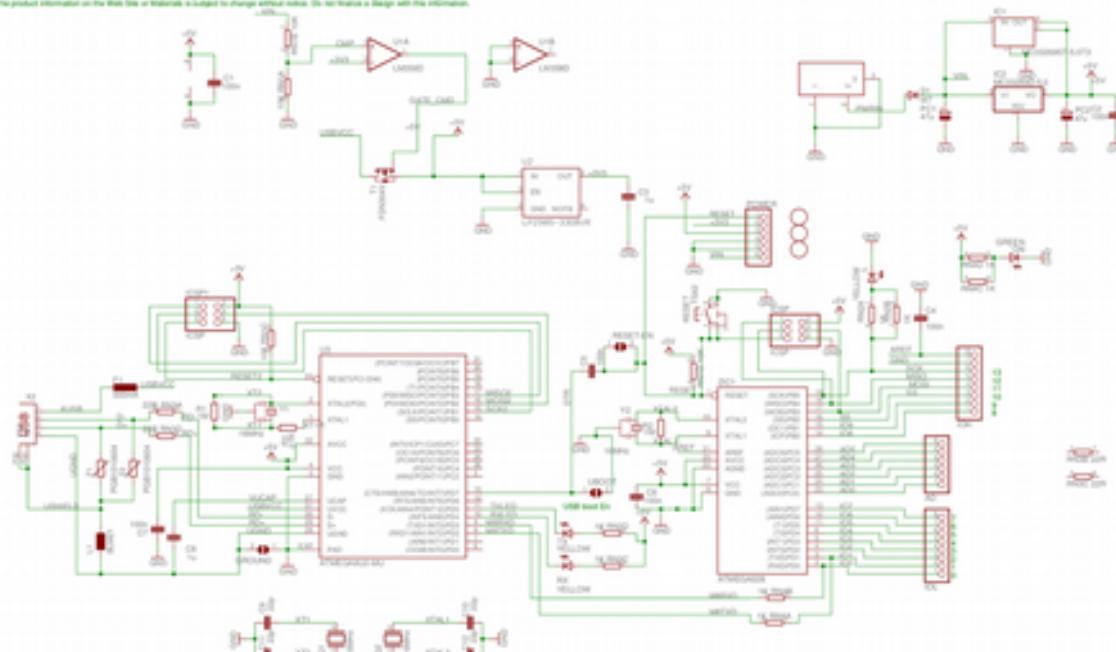


ATMEGA328 PINOUT

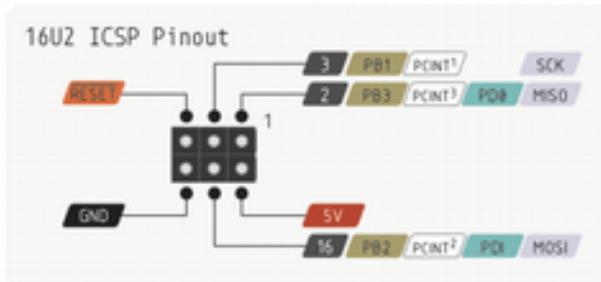
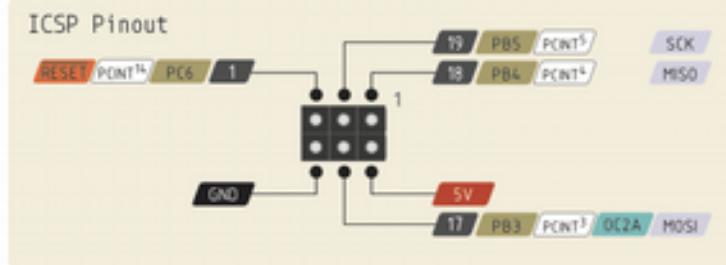
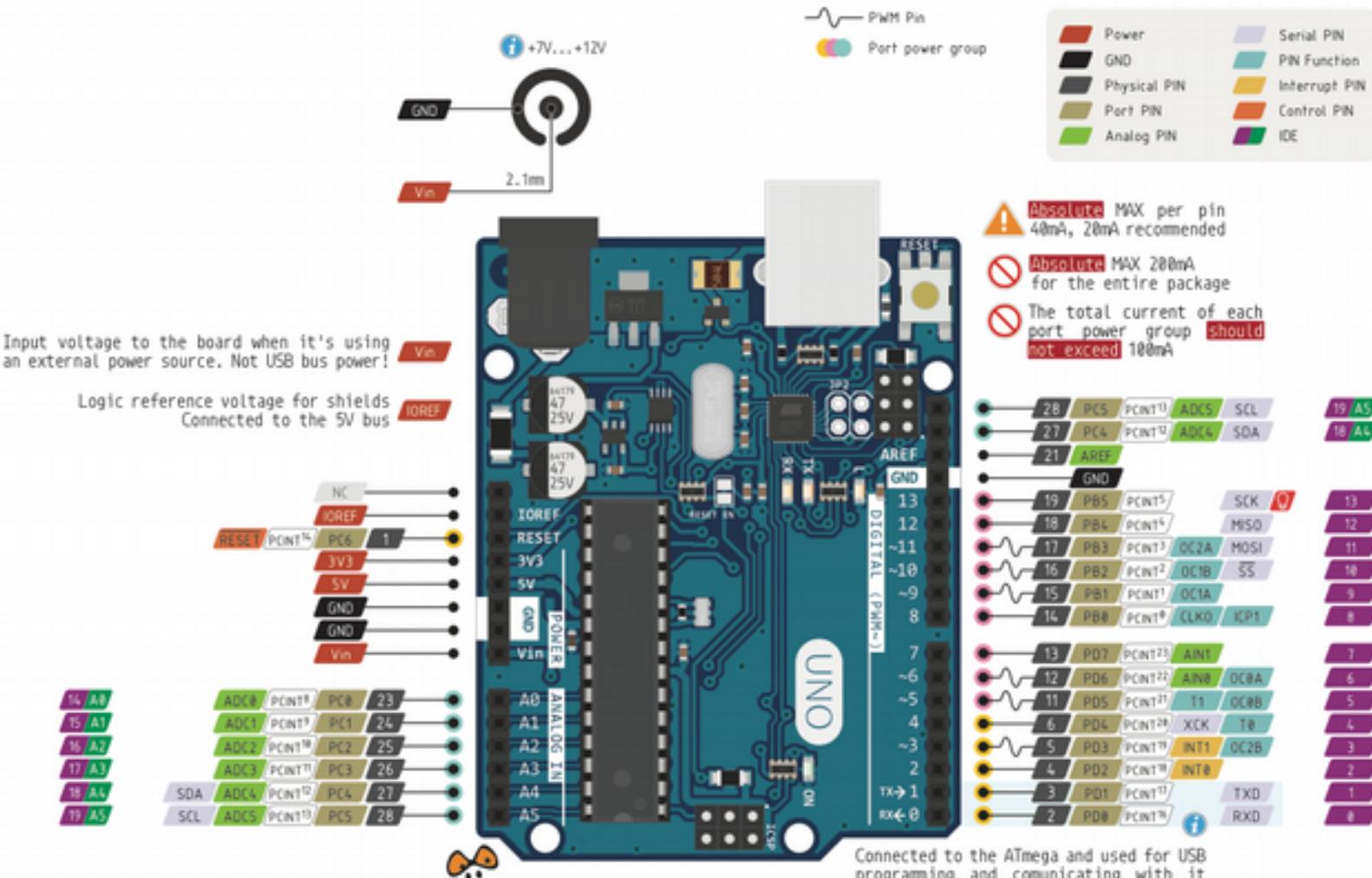


Arduino™ UNO Reference Design

Reference Designs ARE PROVIDED "AS IS" AND WITH ALL FAULTS. ARDUINO DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the information contained in this document as a specification of the product, which may change at any time without notice. The Customer must refer to the latest version of the document or other documentation or publications for future definition and shall keep no responsibility whatsoever for conflicts or inconveniences arising from future changes to them. The product information on this site is subject to change without notice. Do not base a design with this information.

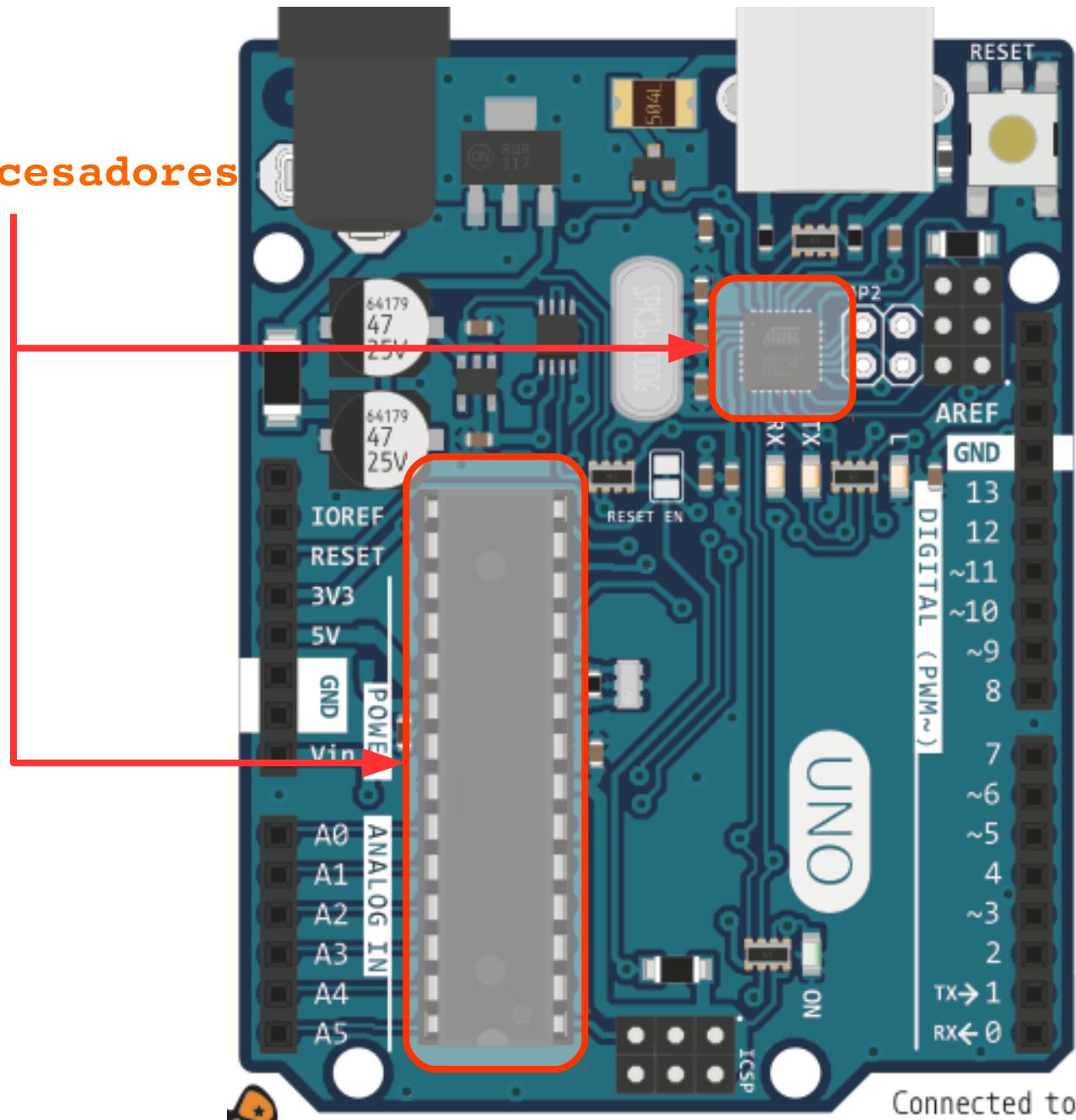


# Hardware: El microprocesador y algo mas...

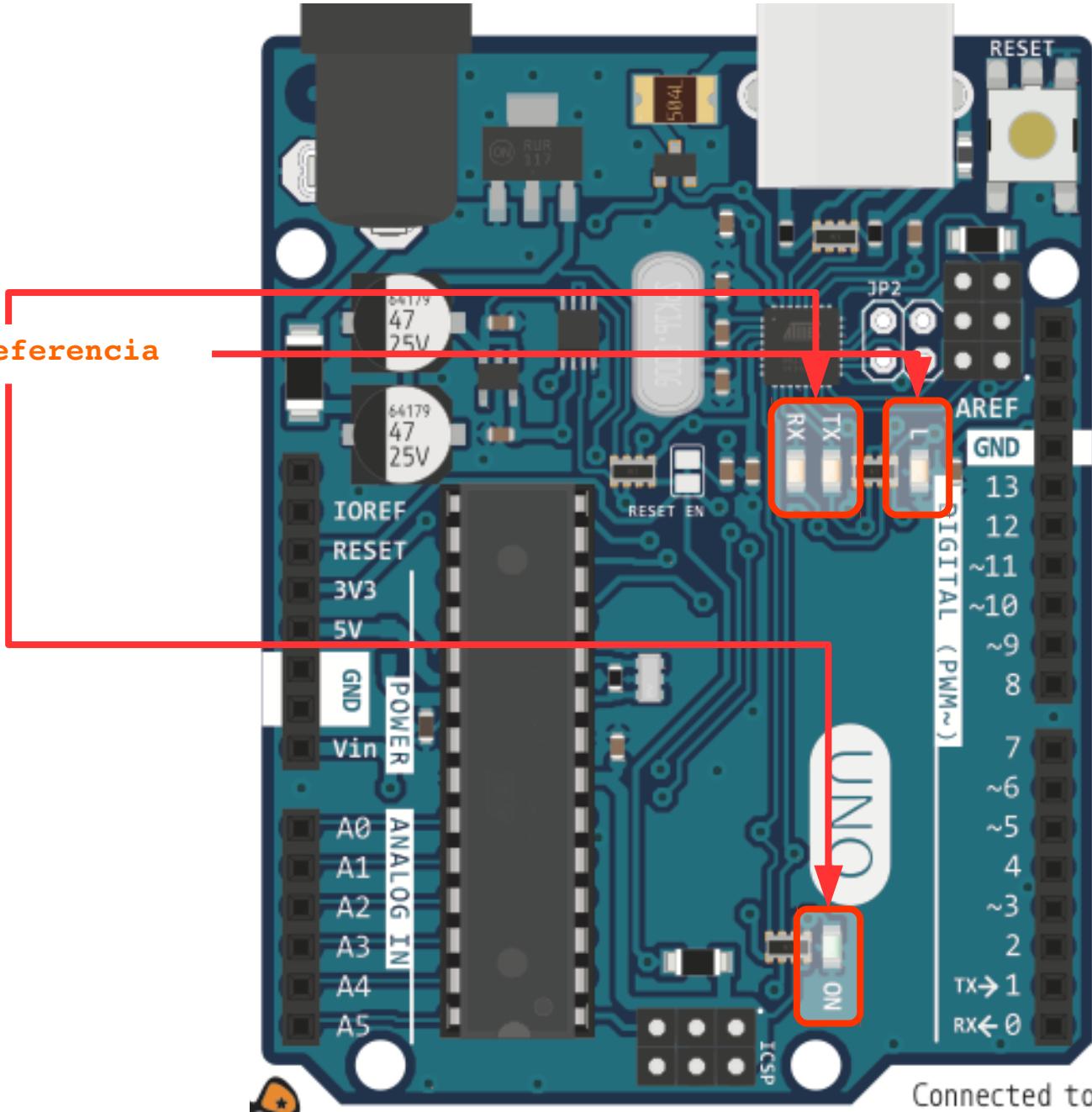


# Hardware: El microprocesador y algo mas...

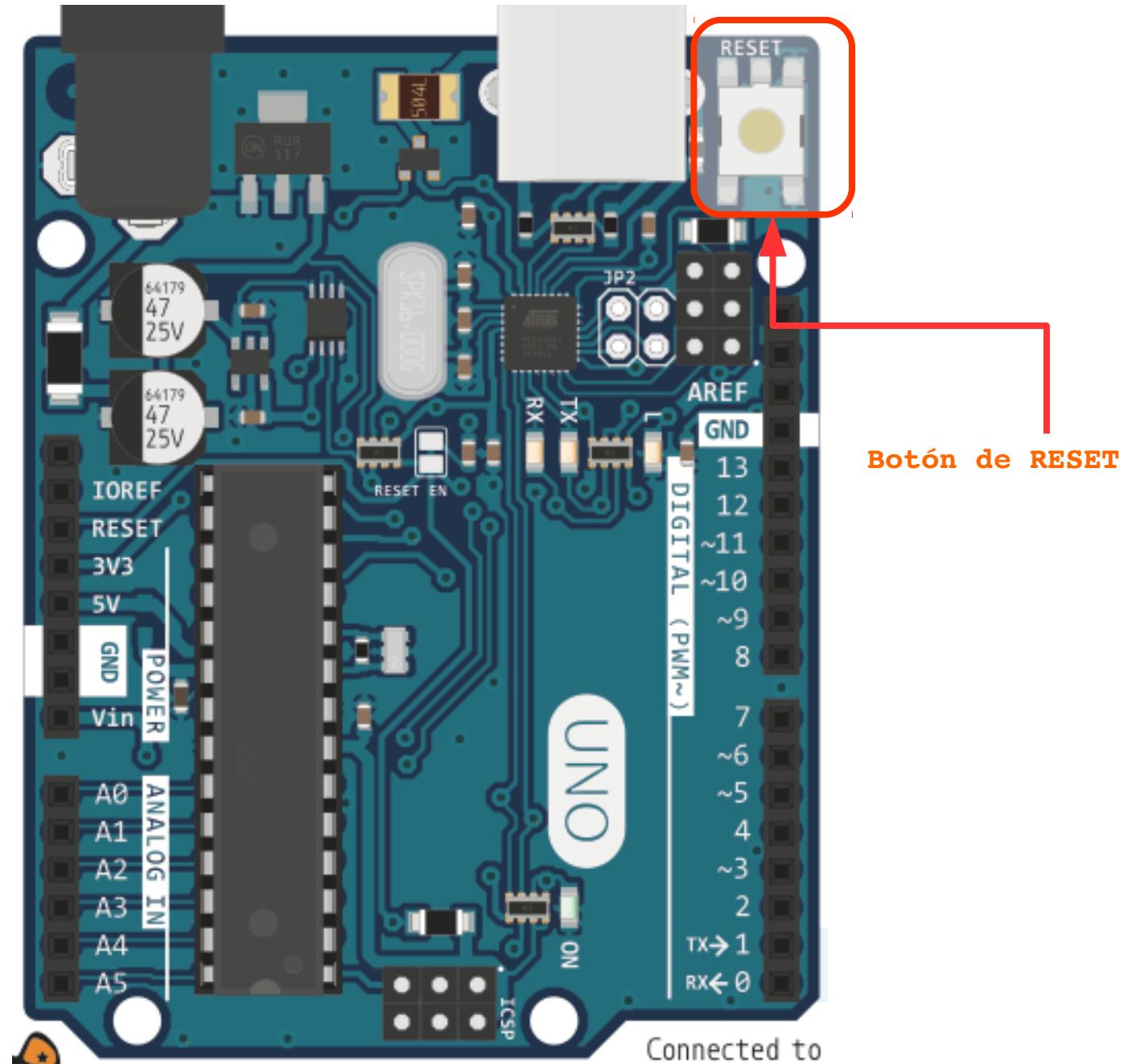
microprocesadores



# Hardware: El microprocesador y algo mas...

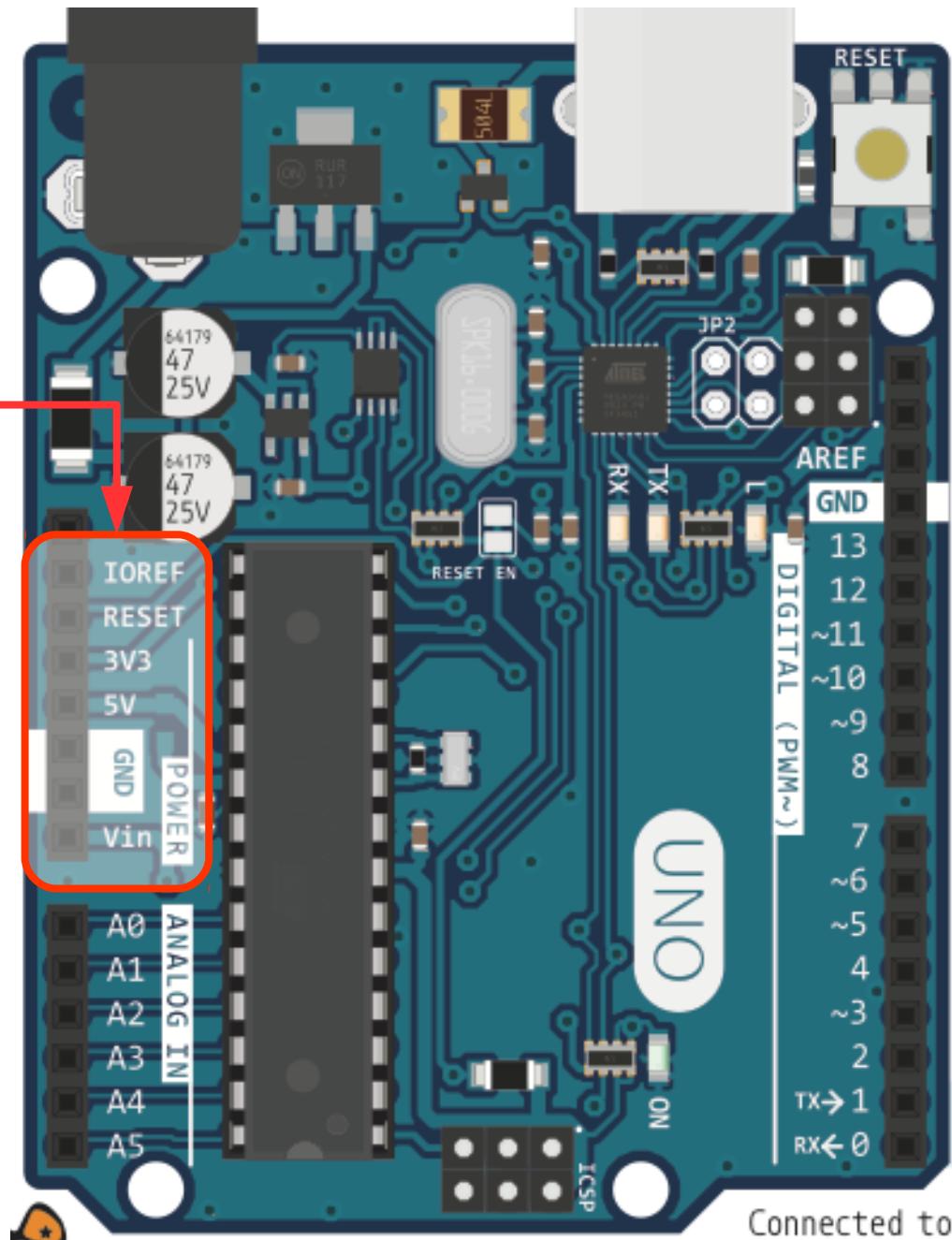


# Hardware: El microprocesador y algo mas...

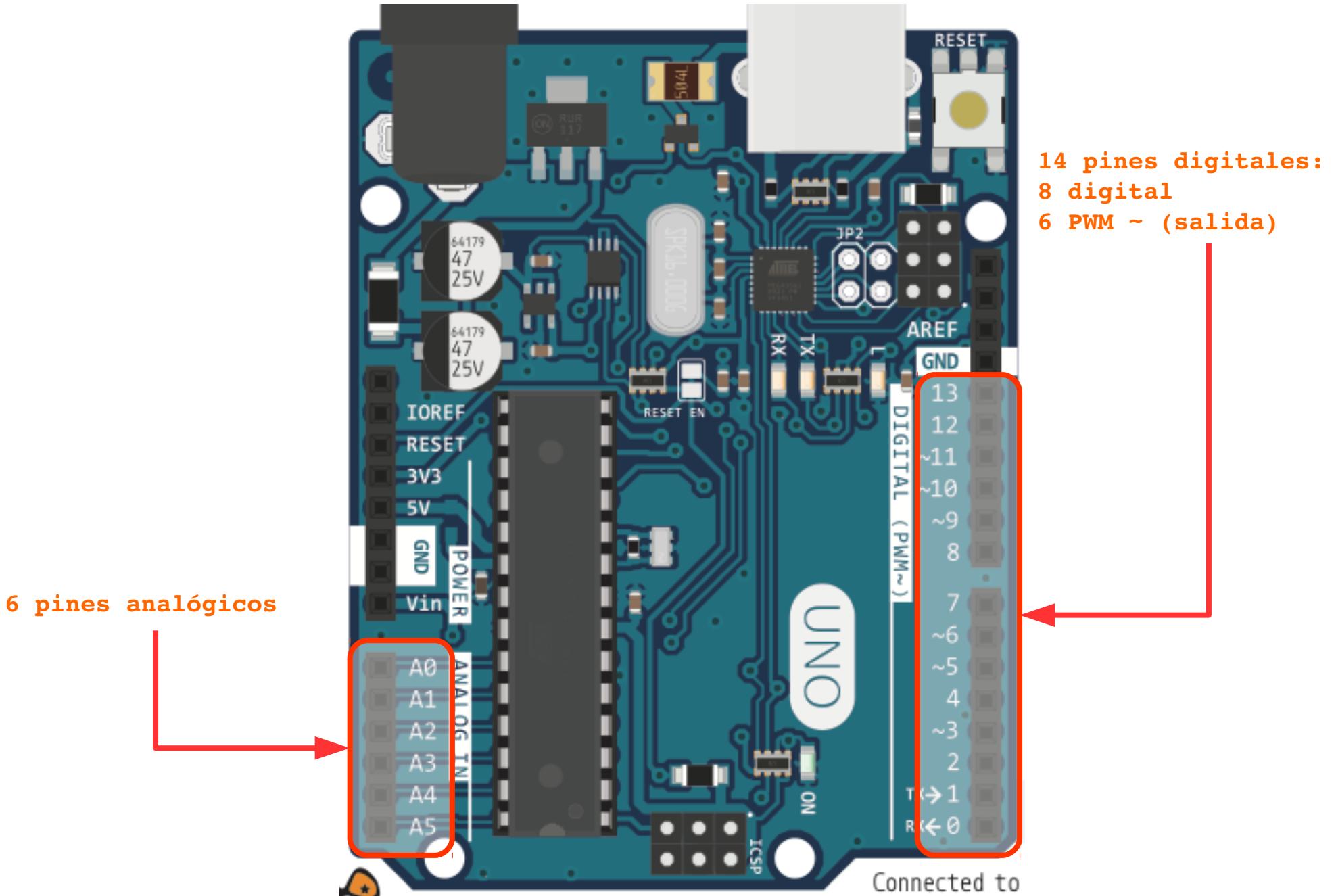


# Hardware: El microprocesador y algo mas...

Pines de voltaje

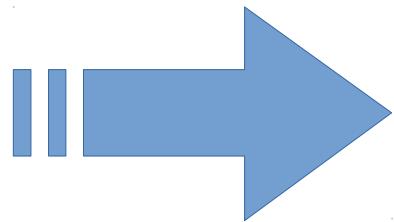


# Hardware: El microprocesador y algo mas...

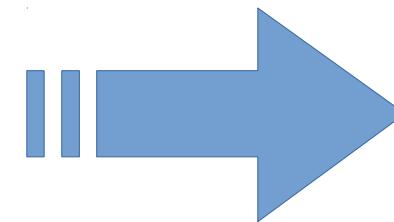


# Software

# Software: Antepasados comunes del lenguaje



<https://processing.org/>



<http://wiring.org.co/>



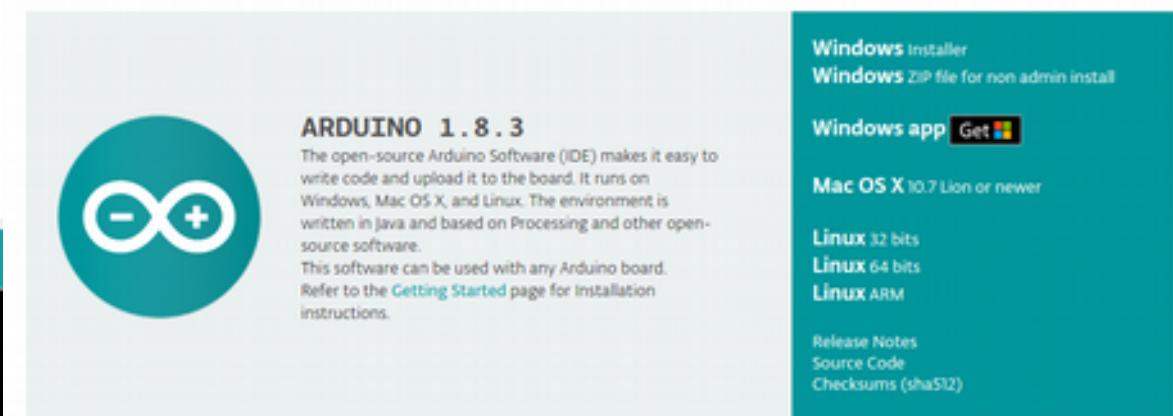
# Software: IDE. El entorno para humanos

```
sketch_jul09a Arduino 1.8.2
Archivo Editar Programa Herramientas Ayuda
sketch_jul09a.ino
void setup() {
  // put your setup code here, to run once
}

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



Download the Arduino IDE



# LENGUAJE HUMANO . Estructura básica de UN PROGRAMA

Librerías

Variables Globales

Constantes

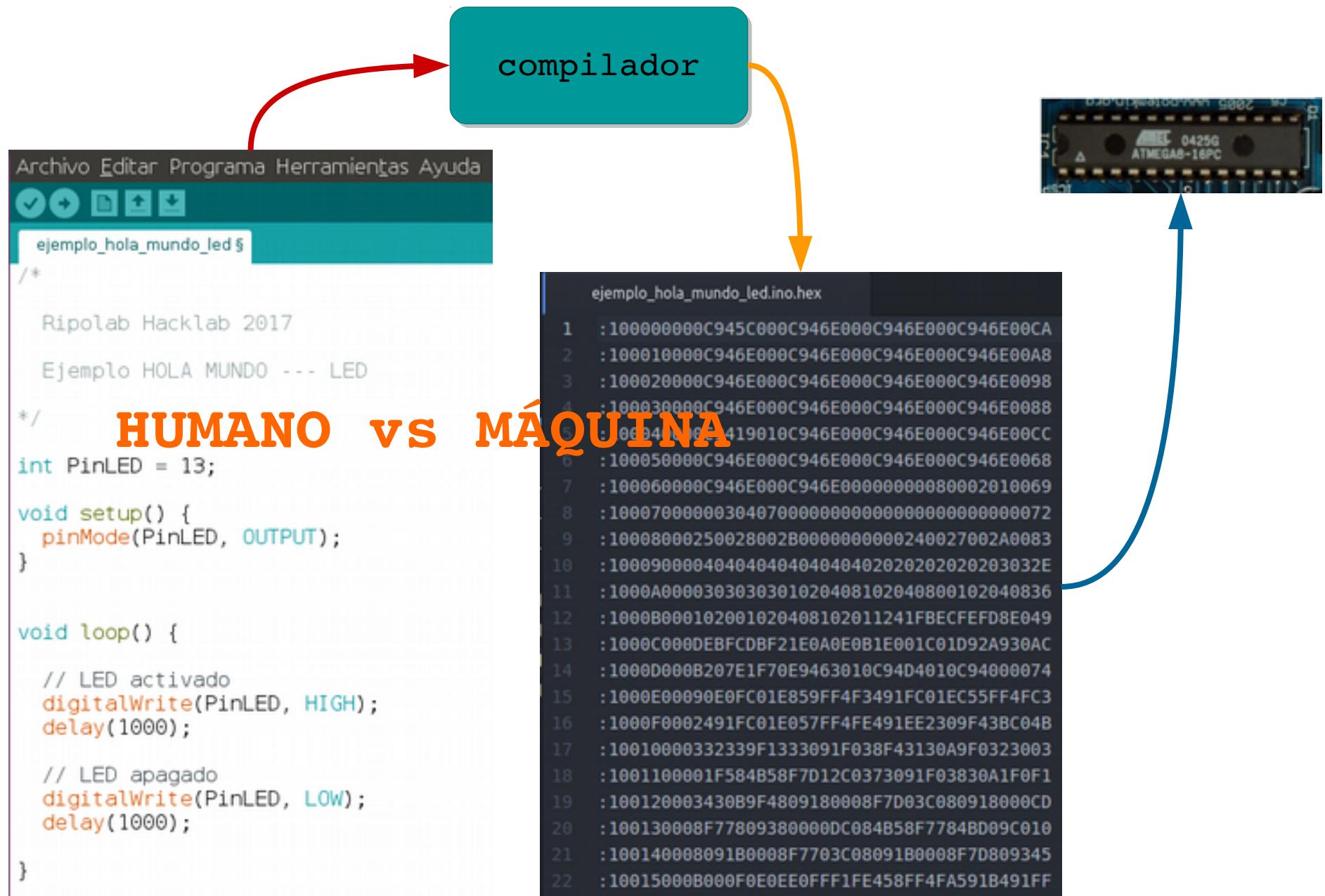
Funciones propias

```
void setup() {  
    // código de configuración  
}
```

```
void loop() {  
    // código que ejecuta  
    // "para siempre"  
}
```



# COMPILADOR. El traductor: HUMANO vs MÁQUINA



## Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

### Structure

- `setup()`
- `loop()`

#### Control Structures

- `if`
- `if...else`
- `for`
- `switch case`
- `while`
- `do...while`
- `break`
- `continue`
- `return`
- `goto`

#### Further Syntax

- `;` (semicolon)
- `{}` (curly braces)
- `//` (single line comment)
- `/* */` (multi-line comment)
- `#define`
- `#include`

#### Arithmetic Operators

- `=` (assignment operator)
- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

#### Comparison Operators

- `==` (equal to)
- `!=` (not equal to)
- `<` (less than)
- `>` (greater than)
- `<=` (less than or equal to)
- `>=` (greater than or equal to)

### Variables

#### Constants

- `HIGH` | `LOW`
- `INPUT` | `OUTPUT` | `INPUT_PULLUP`
- `LED_BUILTIN`
- `true` | `false`
- integer constants
- floating point constants

#### Data Types

- `void`
- `boolean`
- `char`
- `unsigned char`
- `byte`
- `int`
- `unsigned int`
- `word`
- `long`
- `unsigned long`
- `short`
- `float`
- `double`
- `string` - char array
- `String` - object
- `array`

#### Conversion

- `char()`
- `byte()`
- `int()`
- `word()`
- `long()`
- `float()`

#### Variable Scope & Qualifiers

- `variable scope`
- `static`
- `const`

### Functions

#### Digital I/O

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

#### Analog I/O

- `analogReference()`
- `analogRead()`
- `analogWrite()` - PWM

#### Advanced I/O

- `tone()`
- `noTone()`
- `shiftOut()`
- `shiftIn()`
- `pulseIn()`

#### Time

- `millis()`
- `micros()`
- `delay()`
- `delayMicroseconds()`

#### Math

- `min()`
- `max()`
- `abs()`
- `constrain()`
- `map()`
- `pow()`
- `sqrt()`

#### Trigonometry

- `sin()`

### Boolean Operators

- `&&` (and)
- `||` (or)
- `!` (not)

### Pointer Access Operators

- `*` dereference operator
- `&` reference operator

### Bitwise Operators

- `&` (bitwise and)
- `||` (bitwise or)
- `^` (bitwise xor)
- `~` (bitwise not)
- `<<` (bitshift left)
- `>>` (bitshift right)

### Compound Operators

- `++` (increment)
- `--` (decrement)
- `+=` (compound addition)
- `-=` (compound subtraction)
- `*=` (compound multiplication)
- `/=` (compound division)
- `%=` (compound modulo)
- `&=` (compound bitwise and)
- `|=` (compound bitwise or)

### Memory

- `const`

### Utilities

- `sizeof()`
- `PROGMEM`

### Math

- `cos()`
- `tan()`

### Characters

- `isAlphaNumeric()`
- `isAlpha()`
- `isAscii()`
- `isWhitespace()`
- `isControl()`
- `isDigit()`
- `isGraph()`
- `isLowerCase()`
- `isPrintable()`
- `isPunct()`
- `isSpace()`
- `isUpperCase()`
- `isHexadecimalDigit()`

### Random Numbers

- `randomSeed()`
- `random()`

### Bits and Bytes

- `lowByte()`
- `highByte()`
- `bitRead()`
- `bitWrite()`
- `bitSet()`
- `bitClear()`
- `bit()`

### External Interrupts

- `attachInterrupt()`
- `detachInterrupt()`

### Interrupts

- `interrupts()`
- `noInterrupts()`

### Communication

- `Serial`
- `Stream`

### USB (32u4 based boards and Due/Zero only)

- `Keyboard`
- `Mouse`

# Lenguaje Básico. Hablar con las máquinas

**ESCRIBIR / HABLAR**

```
pinMode(#pin, OUTPUT)
```

```
digitalWrite(#pin, HIGH)
```

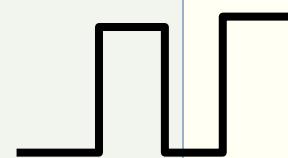
```
digitalWrite(#pin, LOW)
```

```
analogWrite(#pin, #valor)
```

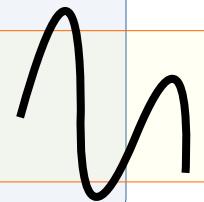
**LEER / ESCUCHAR**

```
pinMode(#pin, INPUT)
```

```
digitalRead(#pin)
```



```
analogRead(#pin)
```



# Lenguaje Básico. ¿Y la pantalla?

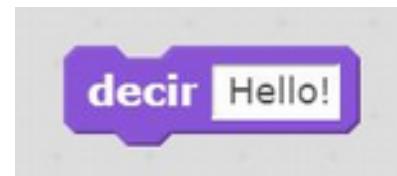


```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println("Hola mundo!!!!");  
}
```

The screenshot shows the Arduino IDE interface. The top menu bar includes Archivo, Editar, Programa, Herramientas, and Ayuda. The title bar says "ejemplo\_hola\_mundo\_led\_serial Arduino 1.8.2". The code editor contains the following sketch:

```
/*  
Ripolab Hacklab 2017  
Ejemplo HOLA MUNDO  
*/  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println("Hola mundo!!!!");  
}
```

The serial monitor window below shows the output: "Hola mundo!!!!" repeated multiple times. The status bar at the bottom indicates "Autoscroll" is checked and the baud rate is set to "9600 bautio".



# Lenguaje Básico. Repetir y repetir



```
for ([valor inicial]; [condición]; [incremento/decremento])  
{  
    // código que se repite  
}
```



# Lenguaje Básico. Tomar decisiones



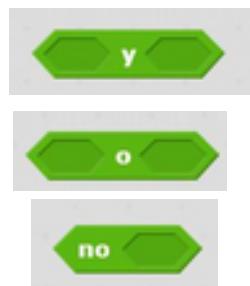
```
if ([condición])
{
    // código que se ejecuta si
    // se cumple la condición_1
}
```



```
if ([condición_1])
{
    // código que se ejecuta si se
    // cumple la condición_1
}

else

{
    // código que se ejecuta si
    // NO se cumple la condición_1
}
```



# Lenguaje Básico. Espera un momento...

```
delay(tiempo milisegundos);
```

```
delayMicroseconds(tiempo microisegundos);
```



# Lenguaje Básico. Condiciones y booleanos

Igual	<code>==</code>	
No Igual	<code>!=</code>	
Menor que	<code>&lt;</code>	
Mayor que	<code>&gt;</code>	
Menor igual que	<code>&lt;=</code>	
Mayor igual que	<code>=&gt;</code>	
Y	<code>&amp;&amp;</code>	
O	<code>  </code>	
NO	<code>!</code>	

**Al cacharreo!!!**

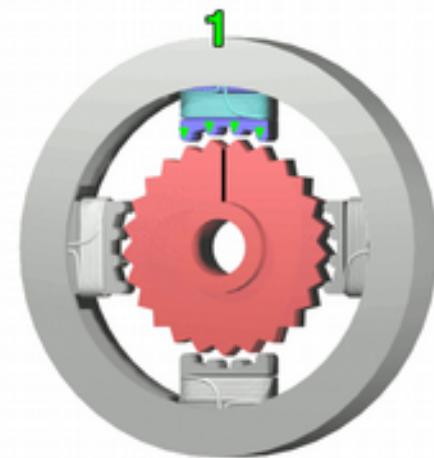
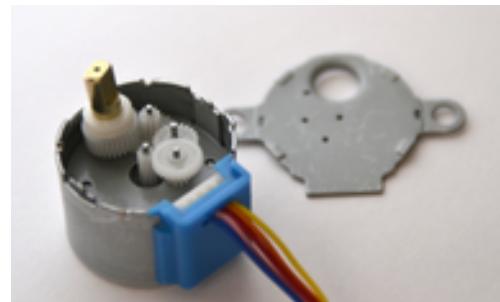
**#RETO 0**

# **Teoría para el #RETO 1**

# #RETO 1. Elementos necesarios

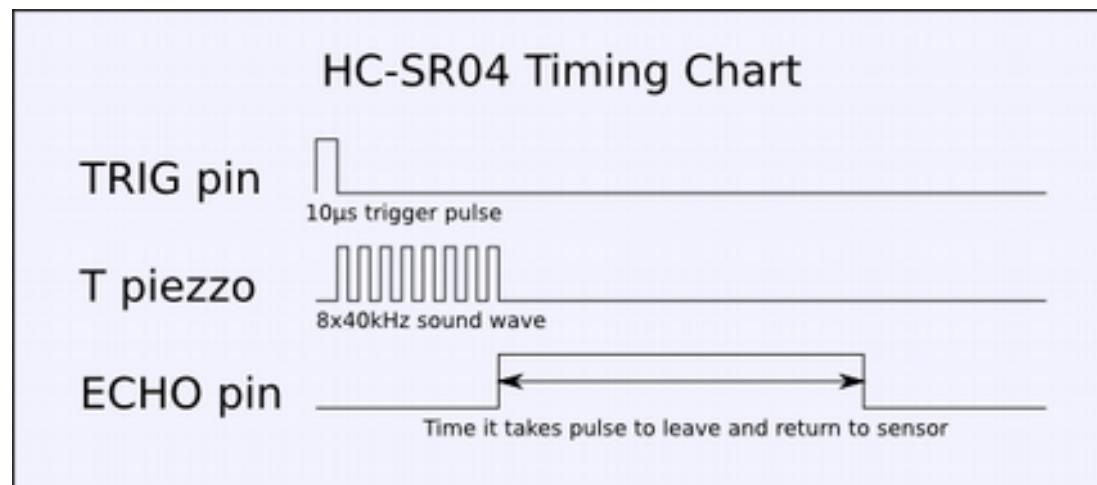
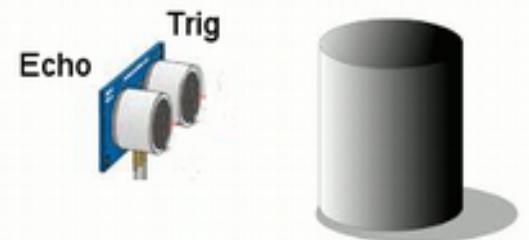


Motor paso a  
paso 28BYJ-48



# #RETO 1. Elementos necesarios

## Sensor Ultrasonido HC-SR04



**Lenguaje Avanzado. Las cosas raras**

**OOP**

**Oriented-Object Programming**

**POO**

**Programación Orientada a Objeto**

# Lenguaje Avanzado. oop / POO



Utilizamos una librería

***Coche.h***

Creo un objeto del tipo Coche

***Coche MiCoche;***

Utilizo un método que me permite definir el color

***MiCoche.color("azul");***

Múltiples métodos:

***MiCoche.encender();***

***MiCoche.apagar();***

***MiCoche.marcha(1);***

***MiCoche.acelerar(100);***

# #RETO 1. Librerías necesarias



```
#include <Stepper.h>
```

```
setSpeed();  
step();
```

<https://www.arduino.cc/en/Reference/Stepper>



```
#include <Ultrasonic.h>
```

```
distanceRead();
```

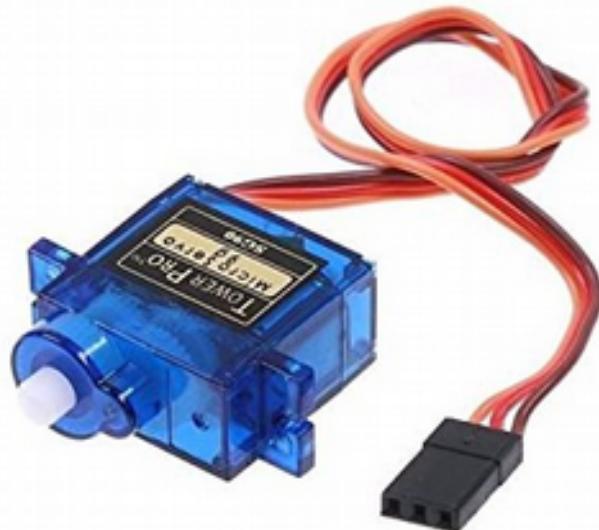
<https://github.com/ErickSimoes/Ultrasonic>

**Al cacharreo!!!**

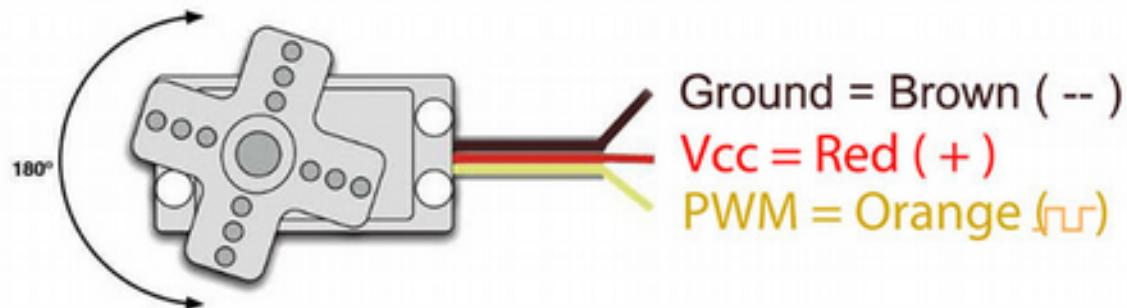
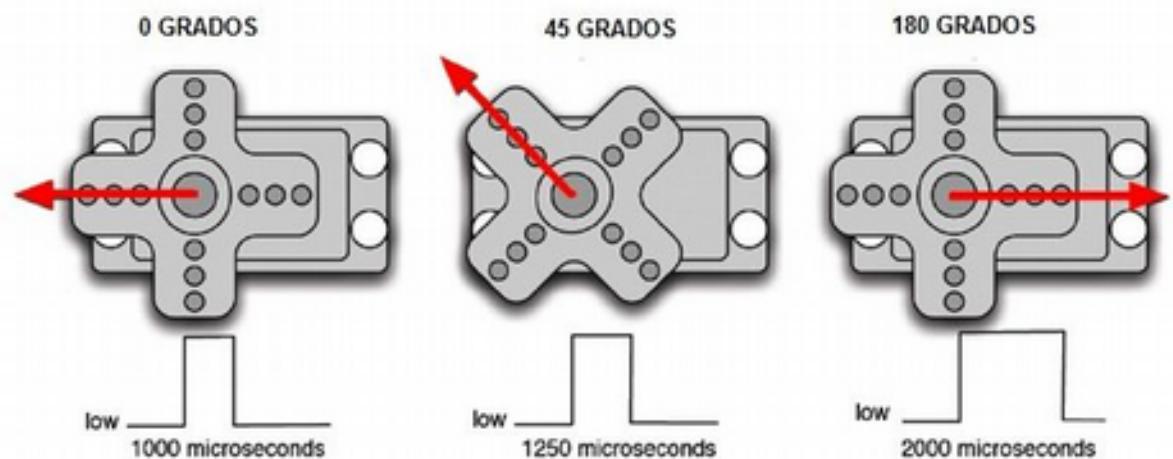
**#RETO 1**

# **Teoría para el #RETO 2**

# #RETO 2. Elementos necesarios



Micro Servo Motores  
**SG90**



# #RETO 2. Librerías necesarias



```
#include <Servo.h>
```

```
Servo Mi_servo;           // Declarar variable  
  
Mi_servo.attach(pin);    // Configurar el pin donde hemos conectado el servo  
  
Mi_servo.write(angulo);  // Iniciar movimiento al "angulo" que indicamos  
  
Mi_servo.read();         // Lee el angulo actual del servo
```

# Lenguaje Avanzado. Bloques de código

## Novato

```
#define PIN_LED 13

void setup() {
    pinMode(PIN_LED, OUTPUT);
}

void loop() {
    digitalWrite(PIN_LED, HIGH);
    delay(1000);
    digitalWrite(PIN_LED, LOW);
    delay(1000);

    digitalWrite(PIN_LED, HIGH);
    delay(1000);
    digitalWrite(PIN_LED, LOW);
    delay(1000);

    digitalWrite(PIN_LED, HIGH);
    delay(250);
    digitalWrite(PIN_LED, LOW);
    delay(250);

    digitalWrite(PIN_LED, HIGH);
    delay(250);
    digitalWrite(PIN_LED, LOW);
    delay(250);
}
```

## Avanzado

```
#define PIN_LED 13

void parpadeo_lento() {
    digitalWrite(PIN_LED, HIGH);
    delay(1000);
    digitalWrite(PIN_LED, LOW);
    delay(1000);
}

void parpadeo_rapido() {
    digitalWrite(PIN_LED, HIGH);
    delay(250);
    digitalWrite(PIN_LED, LOW);
    delay(250);
}

void setup() {
    pinMode(PIN_LED, OUTPUT);
}

void loop() {
    parpadeo_lento();
    parpadeo_lento();
    parpadeo_rapido();
    parpadeo_rapido();
}
```

## Ninja Code

```
#define PIN_LED 13

void parpadeo(int pausa) {
    digitalWrite(PIN_LED, HIGH);
    delay(pausa);
    digitalWrite(PIN_LED, LOW);
    delay(pausa);
}

void setup() {
    pinMode(PIN_LED, OUTPUT);
}

void loop() {
    parpadeo(1000);
    parpadeo(1000);
    parpadeo(250);
    parpadeo(250);
}
```

**Al cacharreo!!!**

**#RETO 2**