PRAKTIKUM UPV 2017

¡BIENVENIDOS!

INTRODUCCIÓN A ARDUINO

ÍNDICE

1. ¿QUÉ ES PROGRAMAR?

- Es la acción de escribir una serie de instrucciones en un lenguaje determinado para que algo o alguien lo ejecute

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
 <head>
  <title>sample</title>
 </head>
 <body>
  <p>Voluptatem accusantium
   totam rem aperiam.</p>
 </body>
</html>
```

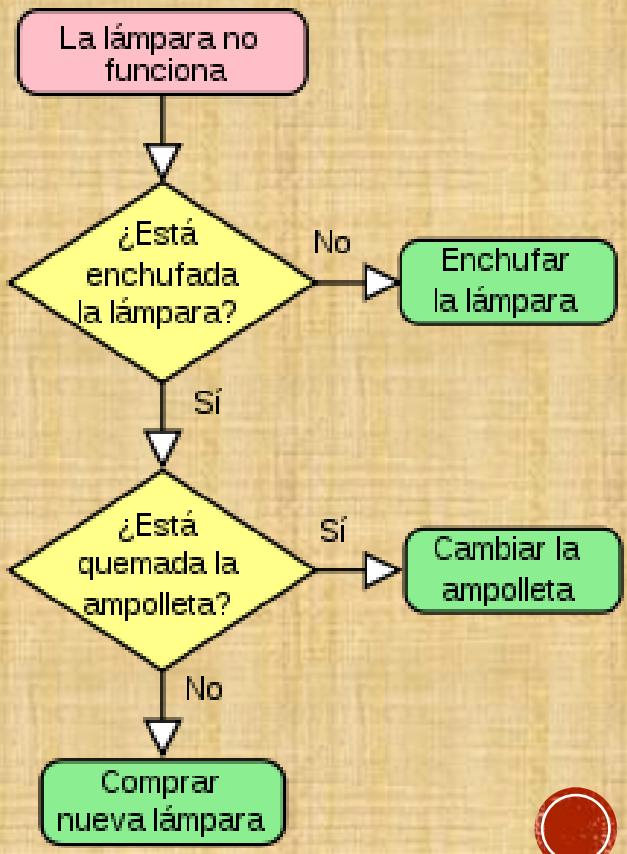
HTML

C/C++

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

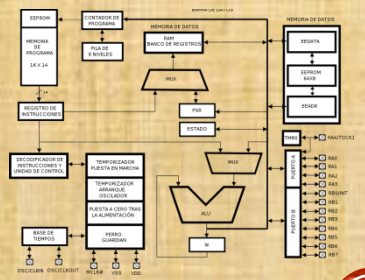


2. LENGUAJES DE PROGRAMACIÓN

Alto nivel

Medio nivel

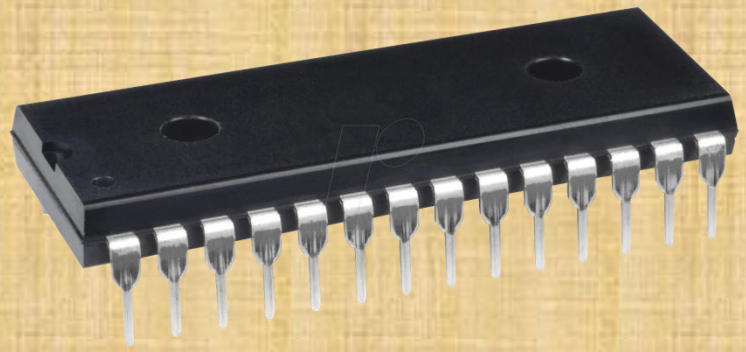
Bajo nivel

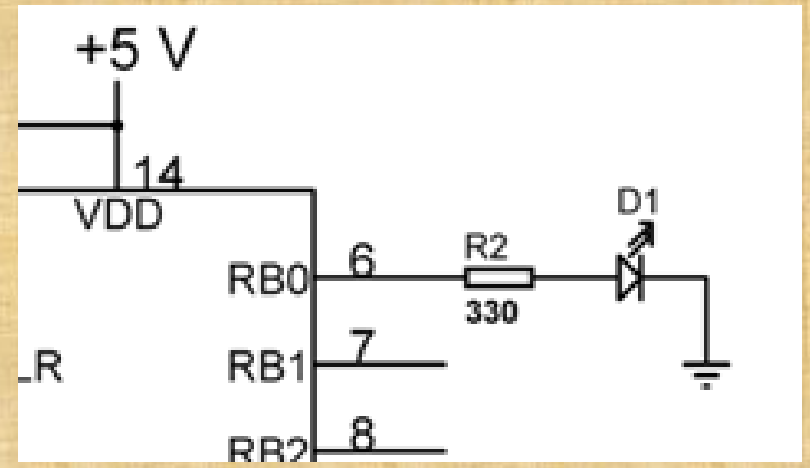


3. EL MICROCONTROLADOR

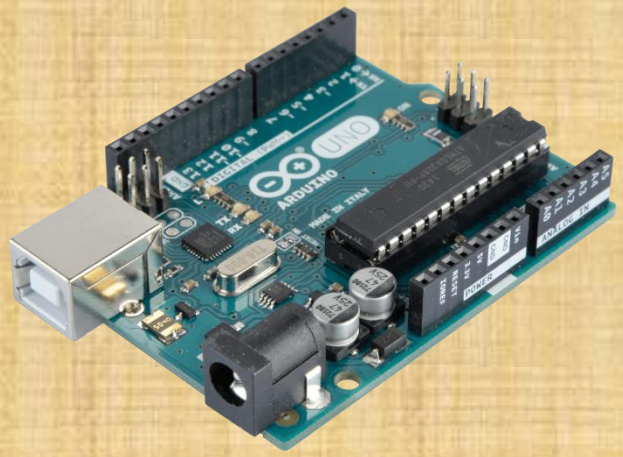
- Es un circuito integrado capaz de ejecutar ordenes que han sido programadas en su memoria.

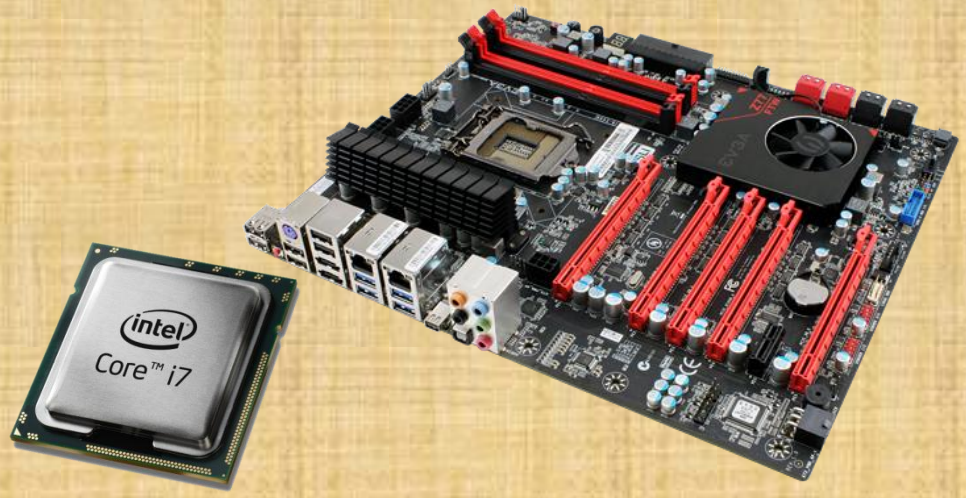
- Es contiene en miniatura todos los elementos mínimos que poseería un ordenador



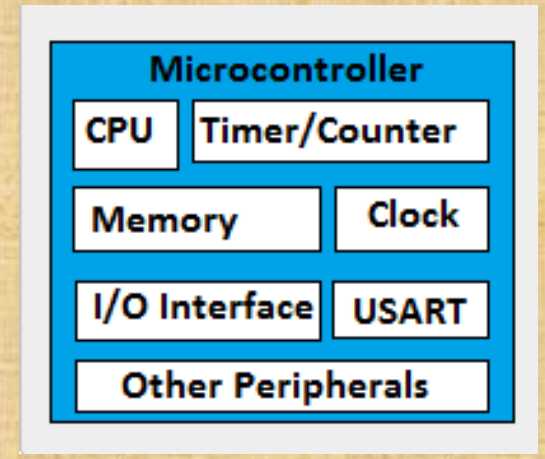


3.1. DIFERENCIAS ENTRE UP Y UC

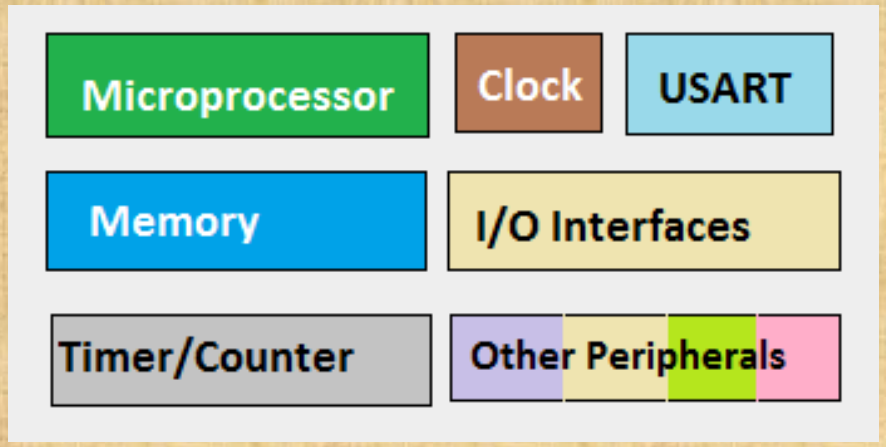




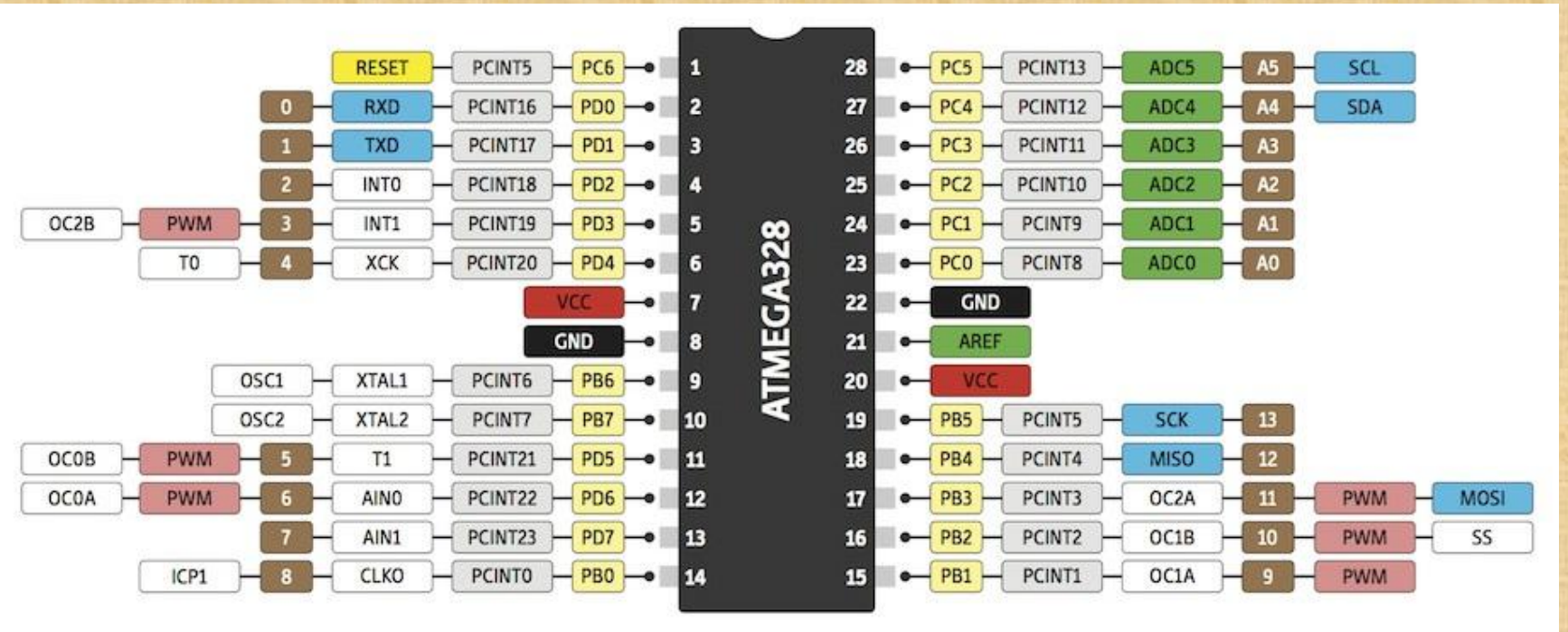
Microcontrolador

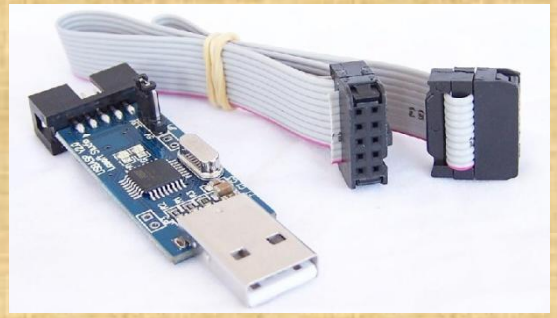


Microprocesador

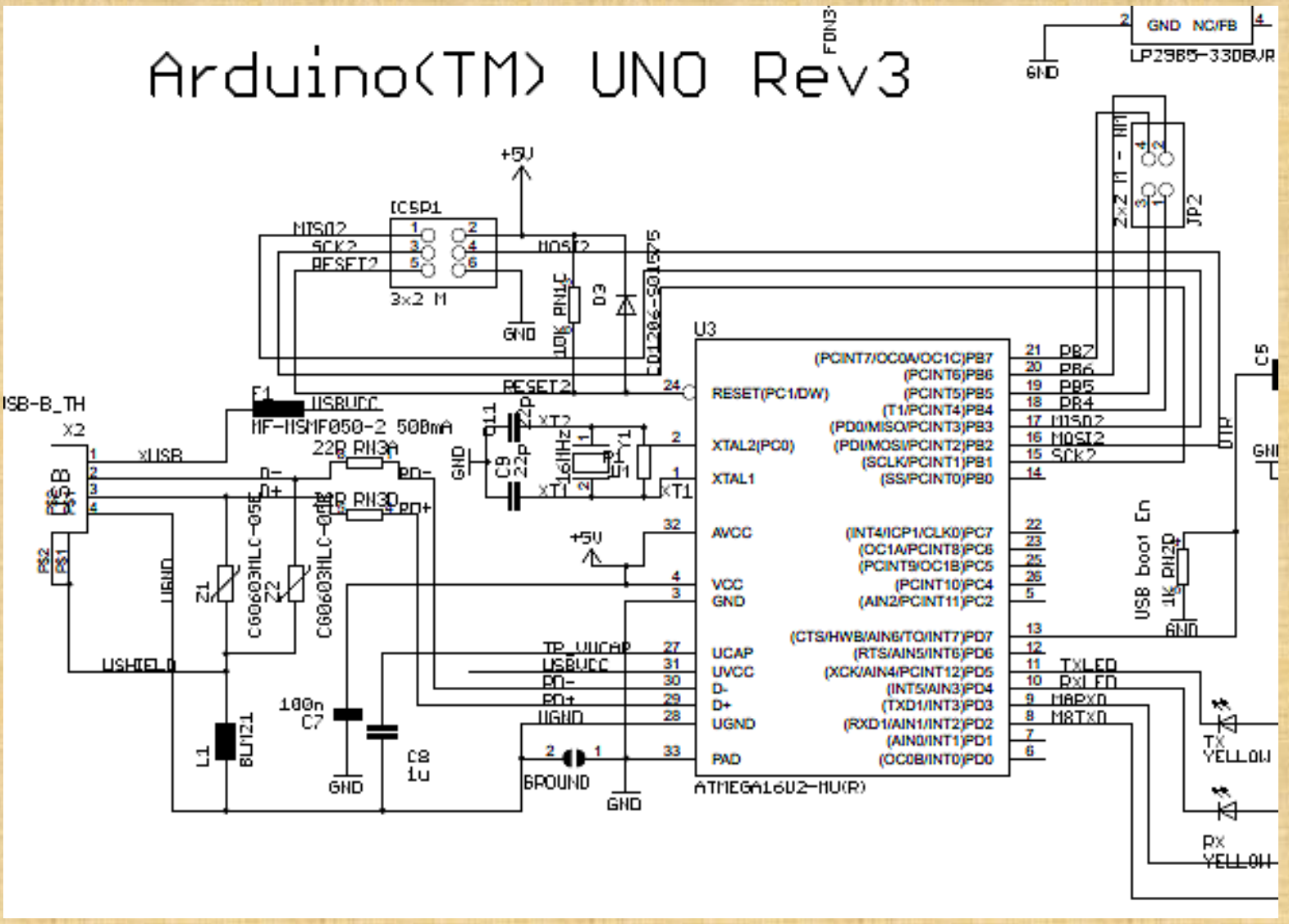


3.2. ATMEGA328P





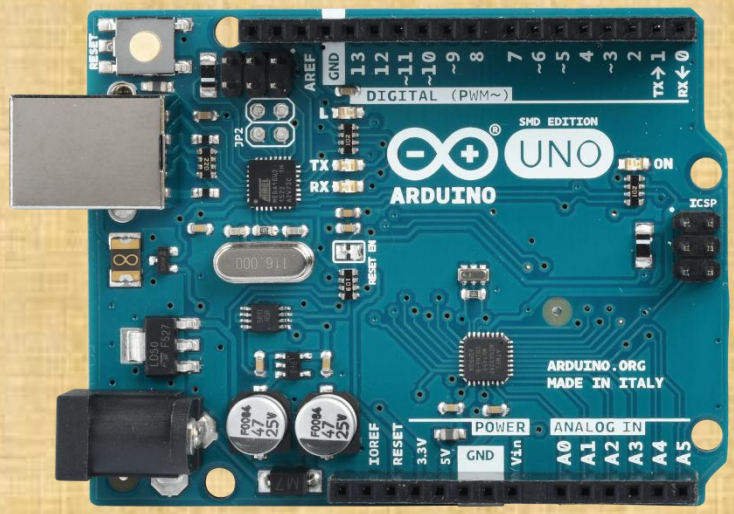
3.2. ATMEGA328P



3.3. ARDUINO UNO

Nuestro Arduino uno usa uno llamado Atmega328P-AU.

El microcontrolador es el cuadrado que hay más o menos en el centro. Que, aunque parezca complicado, trabaja como un enorme pasillo de 28 puertas.

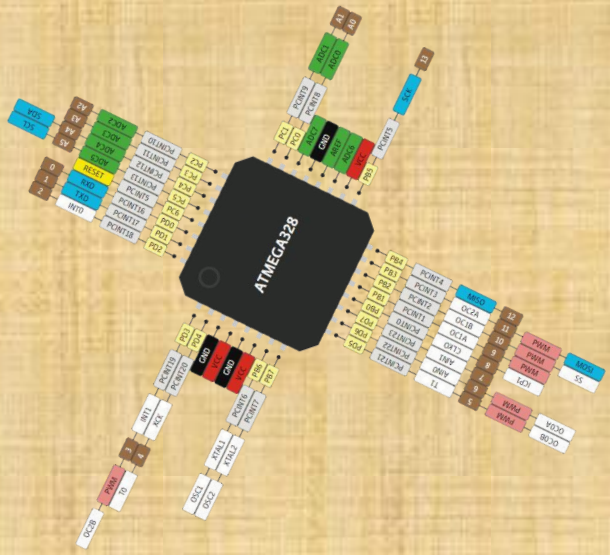


Cada puerta puede ser una entrada o una salida (INPUT, OUTPUT).

Una puerta puede comunicarse con otra si así lo programamos.

Tenemos una, llamada RESET, que reinicia el Arduino

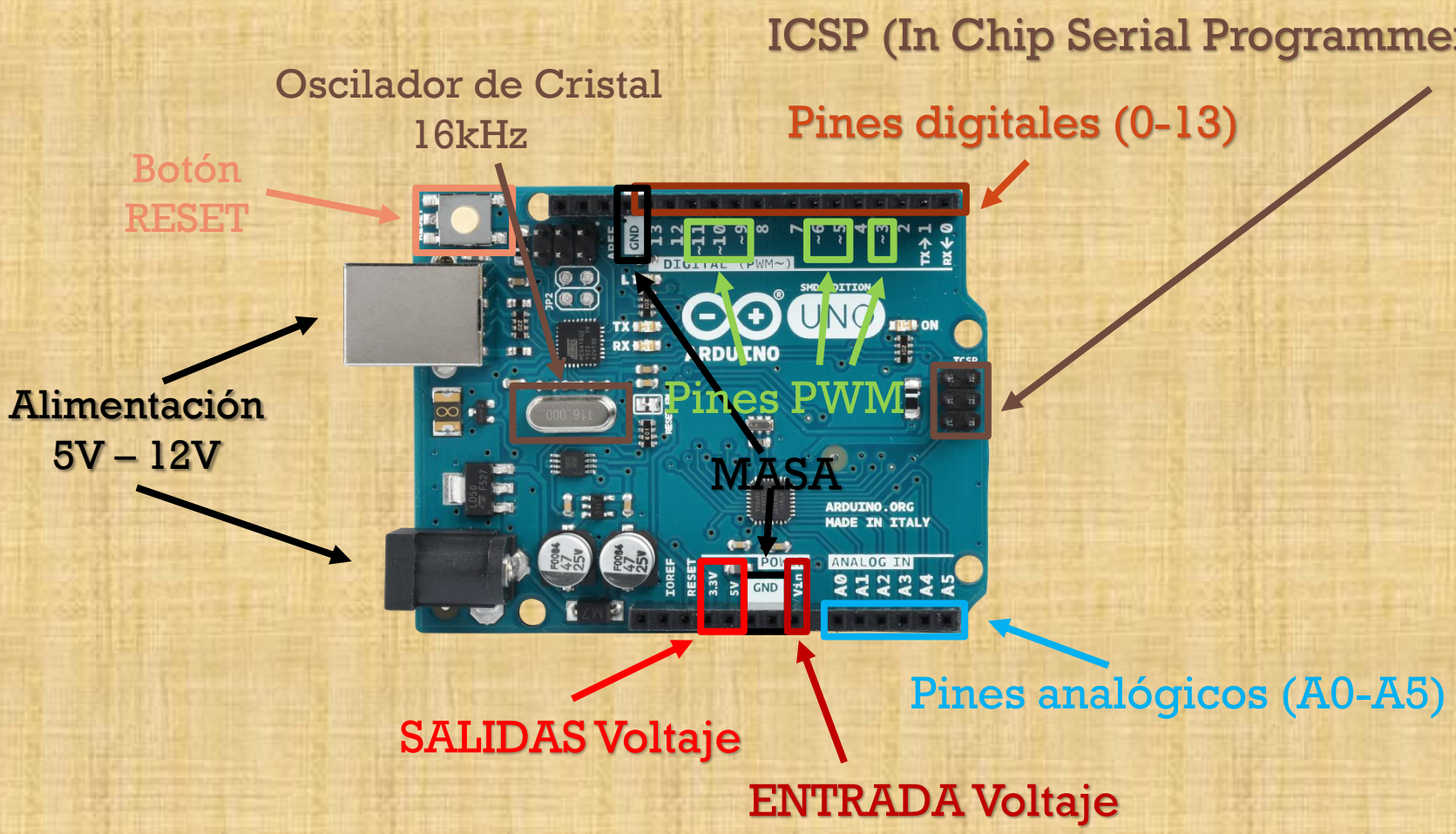
Otras son entradas o salidas de voltaje, sin olvidarnos de Masa (GND).



3.3. ARDUINO UNO

NUESTRO ARDUINO:

- **Arduino UNO Rev 3**



3.3. ARDUINO UNO

¿Y qué características tienen las placas Arduino UNO?

Microcontrolador	ATmega328P
Voltaje operativo	5V
Alimentación (recomendada)	7-12V
Alimentación (límite)	6-20V
Pines Digitales I/O	14 (de los cuales 6 tiene salida PWM)
Pines Digitales PWM I/O	6
Pines analógicos de entrada	6
CC por cada pin I/O	20 mA
CC para pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328P) de los cuales 0.5KB son usados para bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad de Reloj	16 MHz
LED_BUILTIN	13
Longitud	68.6 mm
Anchura	53.4 mm
Peso	25 g

4 INTRODUCCIÓN A C

```
1  // Variables
2
3  void setup() {
4
5    //Configuracion de las entradas
6    // y salidas
7    // Se ejecuta una vez
8  }
9
10 void loop(){
11
12   //Programa principal
13   //Bucle infinito
14
15 }
```

4 INTRODUCCIÓN A C

Tipos de Datos	Memoria que ocupa	Rango de valores
boolean	1 byte	0 o 1 (True o False)
byte / unsigned char	1 byte	0 – 255
char	1 byte	-128 – 127
int	2 bytes	-32.768 – 32.767
word / unsigned int	2 bytes	0 – 65.535
long	2 bytes	-2.147.483.648 – 2.147.483.647
unsigned long	4 bytes	0 – 4.294.967.295
float / double	4 bytes	-3,4028235E+38 - 3,4028235E+38
string	1 byte + x	Array de caracteres
array	1 byte + x	Colección de variables

```
1 bool AbrirCerrar = 1;
2 int personas = 30000;
3 char myChar = 'A';
4 float temperatura = 23.4;
```

4 INTRODUCCIÓN A C

```
22 if (temperatura >= 30)
23 {
24     // Encender aire
25 }
26 else
27 {
28     // Apagar aire
29 }
```

```
37 for(int i = 0; i<=personas; i++)
38 {
39     //Atender
40 }
```

4 INTRODUCCIÓN A C

```
64   while (personas >= 50)
65   {
66
67     // Hacer cosas
68   }
```

```
43   switch (personas) {
44     case 0:
45               //Persona 1
46       break;
47
48     case 1:
49               //Persona 2
50       break;
51
52     case 2:
53               //Persona 3
54       break;
55
56     case 3:
57               // Persona 4
58       break;
59   }
```

5. PROGRAMAR ARDUINO UNO

Arduino jamás entendería una orden dada por nosotros de cualquier forma. Su idioma es uno bien antiguo, C, la base de muchos idiomas de programación actuales.

En esta parte vamos a aprender de qué va C. Todos los programas beben de este idioma, desde el principio, hasta el final.

1. C, mayor pero con marcha.

- Se trata de un idioma creado en los años 70.
- En todos los idiomas de programación deberemos compilar el código la primera vez que lo usamos para poder ejecutarlo.

  - ¿Qué es compilar?

Imaginemos que nuestro programa es un puzzle. El compilador mira a fondo el puzzle para ver si todas las piezas (líneas) están bien puestas (escritas), si hay alguna que no debería ir donde o como la hemos puesto, parará la compilación para avisarnos. Si todas las piezas (líneas) están bien, el código se ejecutará.

- Encontramos en él conceptos como las variables, los arrays o los operadores. Esto vamos a explicarlo a continuación.

5.1. SALIDA DIGITAL

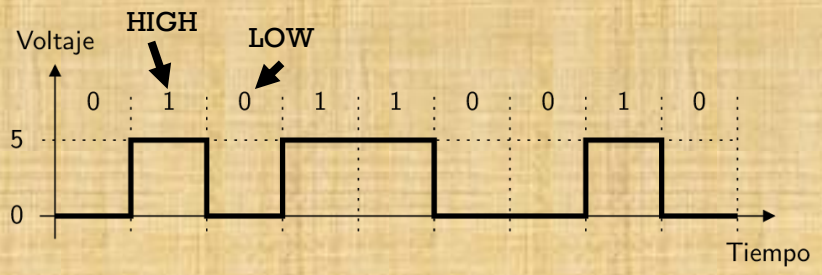
Primero de todo hay que destacar que los pines pueden trabajar como entradas (INPUT), o como salidas (OUTPUT). Esto estará en nuestro código.

1. Pines digitales
   - Estos pines conocen dos valores: HIGH y LOW, podríamos decir: ENCENDIDO y APAGADO (los famosos 1 y 0).
   - En Arduino tenemos un voltaje umbral de 2,5V, todo por encima hasta 5V será HIGH, y todo por debajo de 2,5 será LOW.

Para los pines digitales usaremos digitalWrite(pin,valor) en void loop(), algo que será inicializado mediante pinMode(pin,OUTPUT) en void setup() si es una salida, e INPUT en vez de OUTPUT si se trata de una entrada, como por ejemplo, un botón. Debemos tener cuidado ya que la salida será en 5V.

¡Actividad!

El pin 13 tiene un LED ya soldado en la placa Arduino. Vamos a ver cómo se le da el ritmo.



5.1. ENTRADA DIGITAL

Una entrada digital se puede hacer fácilmente con pulsadores. Como lo haremos con pulsadores, distinguiremos los dos tipos que existen:

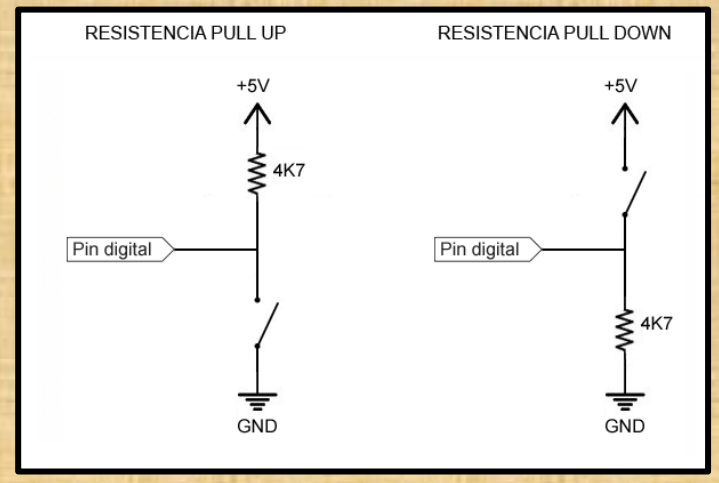
- Resistencia de Pull-up.

El caso que usaremos, consiste en colocar una resistencia entre 5V y el pin (con el pulsador), cuando el pin también está conectado a masa (GND).

Resistencia de Pull-down.

El caso contrario al anterior en el sentido de que la resistencia estará colocada entre masa y el pin con el pulsador. El pin estará además conectado a 5V sin resistencia entre medias.

Definir esto es relativamente sencillo y similar a una salida, en void setup() definiremos pinMode(pin,INPUT) y en void loop(), digitalRead(pin,valor).

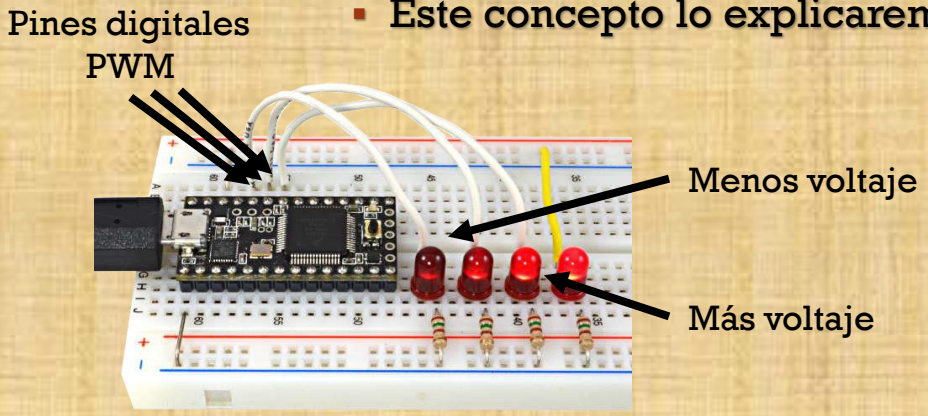


¡Actividad!

Usaremos el mismo pin 13 pero esta vez con un botón pull-up, veamos qué tal se le da.

5.2. SALIDA ANALÓGICA: PWM

1. Pines digitales.
    1. PWM.
        - Son salidas llamadas "pseudo-analógicas", ya que son una versión de estas (las veremos más tarde) pero a la manera digital (es decir, con valores entre 0 y 5V).
        - Dependiendo del valor que le implementemos, programemos o requiramos, este pin registrará o emitirá una señal en este rango de valores.
        - Este concepto lo explicaremos junto con los pines analógicos.



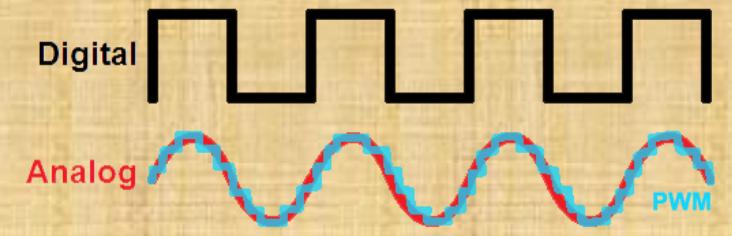
¡Con los pines analógicos veremos de qué son capaces este tipo de pines!

5.3. ENTRADA ANALÓGICA

2. Pines analógicos.
   - Estos pines admiten una variedad de valores que va de 0 a 255, lo cual son 256 valores con los que podemos trabajar.

En el caso de los PWM, 0 serán 0V mientras 255 son 5V. Muchas veces, ambos pines se complementan.

Para usar una señal digital en nuestro void loop() será tan sencillo como usar analogWrite(pin,valor), eso sí, dependiendo de si usamos un pin digital PWM o un analógico deberemos inicializar correctamente en nuestro void setup() el pin correspondiente.



¡Actividad!

Como en la imagen anterior, vamos a ver qué hacen los pines digitales cuando los usamos como PWM usando valores entre 0 y 255. Recuerda: ¡solo 3, 5, 6, 9, 10 y 11!

5.4. ALTERNATIVAS

3. ICSP.
   - Significa "In Chip Serial Programmer".
   - Como sabemos, necesitamos un cable USB para comunicar cualquier chip con el ordenador, estos cabezales no necesitan esta conexión y permiten acceder a su memoria directamente.
   - Arduino puede funcionar como ISP, es decir, puede acceder a la memoria de otros chips directamente y programarlos, sin cable USB.
   - Son esos 6 cabezales a la derecha del Arduino y los nombraremos como el la imagen inferior: