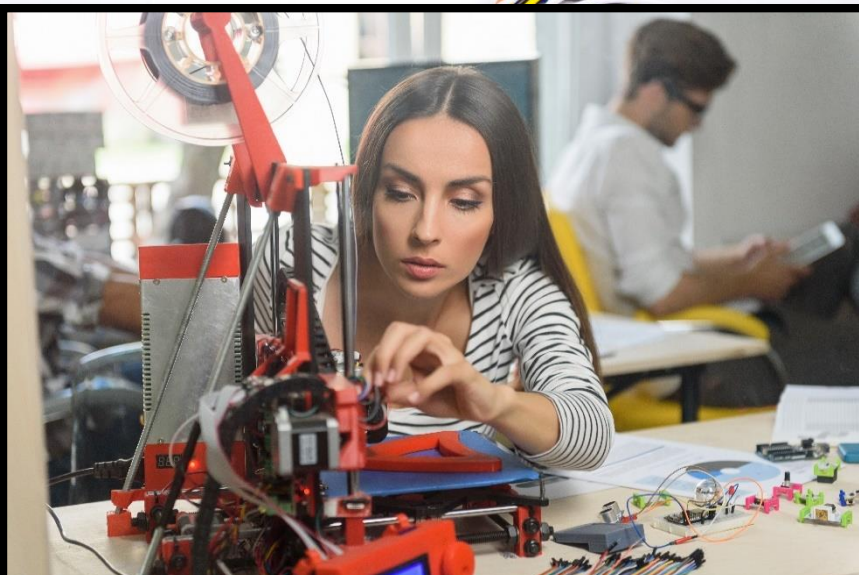
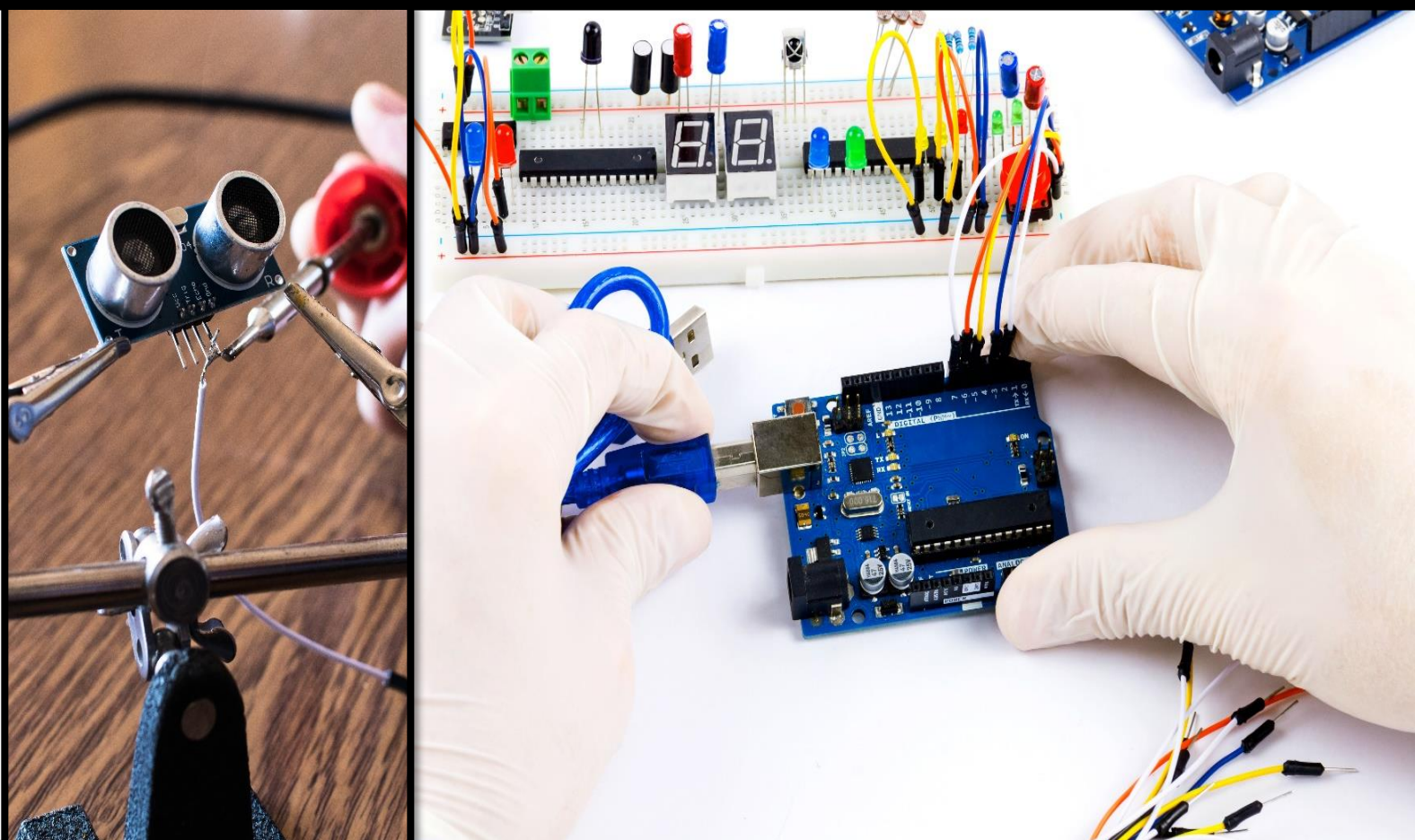
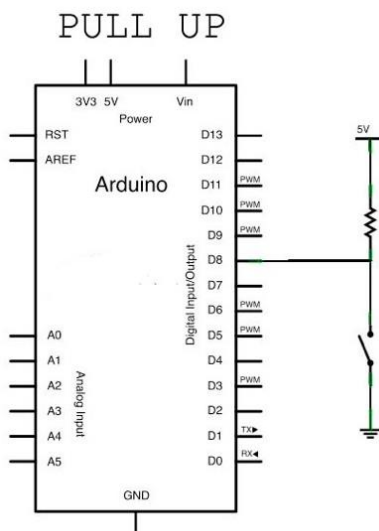


## Resumen Sintaxis Arduino



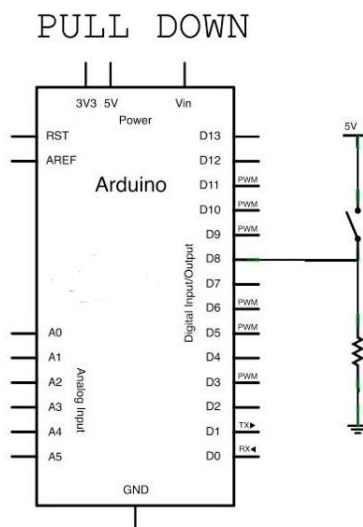
## ENTRADAS

- Conexión mediante resistencia Pull-up: Esta configuración, con una resistencia conectada a la alimentación y la lectura de la entrada en la unión entre la resistencia y el pulsador, establece un nivel lógico de 1 en el estado de reposo del circuito (con el interruptor abierto) y un nivel de 0 cuando se pulsa.









Esta resistencia de *pull-up* puede omitirse si se decide emplear la que incorpora Arduino en sus entradas digitales, para ello se debe indicar durante la programación que la entrada es *pull-up* y se conectará directamente el interruptor a la placa de desarrollo.

- Conexión mediante resistencia Pull-down: Con esta configuración se establece un nivel lógico de 0 en reposo y de 1 al pulsar el interruptor.



Las herramientas más importantes del software IDE que aprenderemos a manejar son las siguientes:

Símbolo	Descripción
	Crear nuevo proyecto
	Abrir un proyecto
	Guardar proyecto
	Compilar y depurar código
	Cargar programa en la placa de Arduino tras compilar
	Abrir la ventana del monitor serie

### Estructura de programa Arduino IDE

#### Variables

```
int a = 0;
int pin = 13;
float pi=3.14;
```

#### setup()

```
pinMode(13, OUTPUT);
pinMode(4, INPUT);
```

#### loop()

```
digitalWrite(13, HIGH);
delay(1000);
```

## Tipos de datos en Arduino

Tipo de variable	Descripción
<b>bool</b>	Variable booleana. Puede tomar valor '0' o '1'
<b>int</b>	Variable de tipo entero. Almacena valores desde -32.767 hasta 32.767
<b>float</b>	Variable de punto flotante. Se emplea para almacenar números decimales
<b>Array</b>	Permite definir vectores de datos. Para ello se debe definir el tipo de dato, el nombre y la extensión. <i>int miArray[5]</i>

## Operadores en Arduino

Operadores Aritméticos	
Operador	Descripción
<b>+/-</b>	Operadores de suma o resta
<b>*</b>	Operador de multiplicación
<b>/</b>	Operador de división
<b>%</b>	Operador que devuelve el resto de una división entera
<b>=</b>	Operador de asignación, permite asignar a la variable que está a la izquierda el valor de la derecha del operador
Operadores de comparación	
<b>!=</b>	Comparación de desigualdad. (Si A es distinto de B)
<b>&lt;/&gt;</b>	Operadores de comparación menor o mayor que
<b>&lt;=</b>	Operador de comparación menor o igual que
<b>==</b>	Operador de comparación de igual. (Si A es igual a B)
Operadores booleanos	
<b>!</b>	Negación lógica
<b>&amp;&amp;</b>	Y lógica (AND)

	O lógica (OR)
--	---------------

Escalado de valores	
<b>map (var, var_min, var_max, desde, hasta)</b>	Escala el valor de la variable a un nuevo rango determinado

## Sintaxis de Arduino

Este lenguaje presenta un conjunto de símbolos e instrucciones que resultan muy útiles.

Símbolo	Descripción
<b><i>/* */</i></b>	Permite realizar un comentario de varias líneas de extensión
<b><i>//</i></b>	Realiza un comentario que finaliza al acabar la línea
<b><i>;</i></b>	Indica el final de una función. Es obligatorio escribirlo al final de cada instrucción o asignación de valores.
<b><i>#define</i></b>	Se escribe antes del <i>setup()</i> y permite definir constantes que serán utilizadas durante todo el programa
<b><i>#include</i></b>	Al igual que <i>#define</i> , se indica antes de comenzar el programa y permite incluir las librerías que contienen las funciones necesarias para nuestro código.

## Principales funciones en Arduino

### Funciones para las entradas y salidas digitales

Función	Descripción
<b><i>pinMode(pin, mode)</i></b>	Configura el pin especificado para comportarse como entrada o como salida. Mediante el primer parámetro se indica el número de pin a configurar y el segundo de ellos el tipo de entrada o salida, pudiendo ser: INPUT, INPUT_PULLUP o OUTPUT.
<b><i>digitalWrite(pin, value)</i></b>	Escribe en la salida digital especificada un valor lógico de estado alto (HIGH) o bajo (LOW).
<b><i>digitalRead(pin)</i></b>	Lee el valor de la entrada digital especificada. Valor lógico HIGH si entrada 5V o LOW si entrada 0V.



## Funciones para las entradas y salidas analógicas

Función	Descripción
<b>analogRead(<i>input</i>)</b>	Lee el valor de tensión entre 0 y 5 voltios de la entrada especificada. Su lectura se realiza en la escala de 0 a 1023 (10 bits).
<b>analogWrite(<i>pin</i>, <i>value</i>)</b>	Carga en el pin seleccionado una señal PWM con un tiempo en alta especificado entre 0 (0%) y 255 (100%).

## Funciones de temporización

Función	Descripción
<b>delay (<i>ms</i>)</b>	Una función de tiempo. Sirve para pausar la ejecución del programa el tiempo especificado (milisegundos)
<b>delayMicroseconds (<i>us</i>)</b>	Función similar a <i>delay()</i> pero el tiempo de pausa es de microsegundos.

## Estructuras de control del Arduino IDE

Sentencia	Descripción
<b>If (<i>condición</i>) / else</b>	Un tipo de control de flujo que ofrece dos o más posibles caminos a tomar por el programa en función de si se cumplen o no las condiciones especificadas.
<b>for(<i>inicio</i>, <i>condición</i>, <i>incremento</i>)</b>	Permite ejecutar un fragmento de código un número determinado de veces de forma cíclica. Se ejecutará mientras la condición no se cumpla e irá incrementándose de la forma que se indique en incremento.
<b>while(<i>condición</i>)</b>	Ejecuta de forma cíclica el conjunto de sentencias especificadas mientras la condición no sea falsa. En el interior del bucle, deberá existir alguna condición que cambie y permita la salida del mismo.
<b>switch... case</b>	Permite ejecutar diferentes secciones de código según el valor de la condición. Cuando el valor coincide con alguno de los casos establecidos, se ejecutará esa parte del código. Al final de cada caso, deberá existir una sentencia de salida <i>break</i> .

A continuación, se muestra un ejemplo de cada estructura.

if/else	
<pre>if (lectura == HIGH){     digitalWrite(9, HIGH); } else{     digitalWrite(9,LOW); }</pre>	<p>Si el valor de la variable lectura es HIGH, se pondrá a 1 la salida del pin 9. En caso contrario, la salida permanecerá desactivada.</p>

for	
<pre>for (int i=0; i&lt;255 ; i++){     analogWrite(10, i);     delay(10); }</pre>	<p>Se realizarán 255 iteraciones, incrementando en cada una de ellas en una unidad el valor de i. En cada una, ese valor se cargará como señal PWM, de modo que el LED que está conectado, irá aumentando su intensidad cada 10 ms.</p>

while	
<pre>var = 0; while (var &lt; 50){     var ++; }</pre>	<p>Se resetea la variable <i>var</i> a 0. Mientras su valor sea menor que 50, se incrementará su valor en una unidad. Cuando este llegue a 50, el bucle <i>while</i> no se ejecutará y la variable será reseteada nuevamente a 0.</p>

switch ... case	
<pre>switch (var){ case 1:     //Sentencias     break; case 2:     //Sentencias     break; default:     //Sentencias     break; }</pre>	<p>Si la variable <i>var</i>, toma el valor 1 se ejecutarán las sentencias escritas entre el <i>case</i> y <i>break</i>. De igual forma sucede con el resto de <i>case</i>.</p> <p>En caso de que la variable no se corresponda con ningún valor se ejecutarán las sentencias del <i>default</i>.</p>



Consejería de Economía,  
Conocimiento y Empleo  
Agencia Canaria de Investigación,  
Innovación y Sociedad  
de la Información

Fondo Europeo de Desarrollo Regional

