INTRODUCCIÓN A ARDUINO

M. C. Felipe Santiago Espinosa

marzo de 2026

2 OBJETIVO DEL TALLER

Que el participante conozca el entorno de desarrollo denominado **Arduino** y programe la tarjeta **Arduino Uno** evaluando el módulo multifunción.

TEMAS A TRATAR

1. Introducción
2. ¿Qué es Arduino?
3. Entorno de desarrollo y programación
4. Salidas Digitales
5. Elementos del lenguaje
6. Entradas digitales
7. Manejo de displays

INTRODUCCIÓN

- Un **microcontrolador** es el "cerebro" en miniatura de un sistema.
- Es un pequeño chip que tiene **todo** lo necesario para **controlar** y hacer **funcionar** cosas específicas, como un robot o un electrodoméstico.
- Dentro de este chip hay una parte que toma decisiones (**el procesador**), otra que guarda información (**la memoria**), y otra que se comunica con los botones, sensores o motores del aparato (**puertos de entrada/salida**).
- Es una **pequeña computadora**, diseñada para hacer **una tarea muy específica** y no muchas al mismo tiempo.

INTRODUCCIÓN

- Un **sistema embebido** es un dispositivo o aparato específico que funciona de manera automática, gobernado por un **microcontrolador**.
- Está **diseñado para realizar tareas concretas**, como controlar el motor de una lavadora, hacer que un termostato ajuste la temperatura o manejar las funciones de un reloj digital.
- **No es como una computadora** que puede hacer muchas cosas diferentes; su trabajo está centrado en **una o pocas tareas**, y normalmente está **integrado dentro de algo más**, como una máquina o un electrodoméstico.

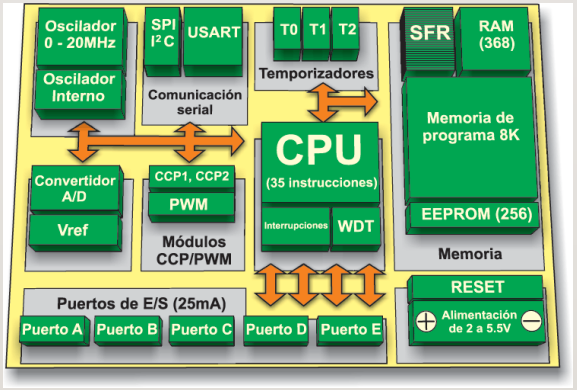
INTRODUCCIÓN

INTRODUCCIÓN

Sistema Tradicional

+

Microcontrolador



=

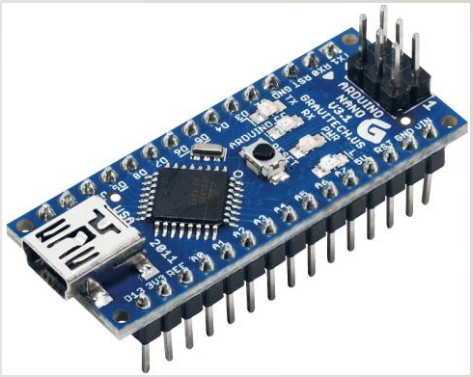
Sistema Embebido

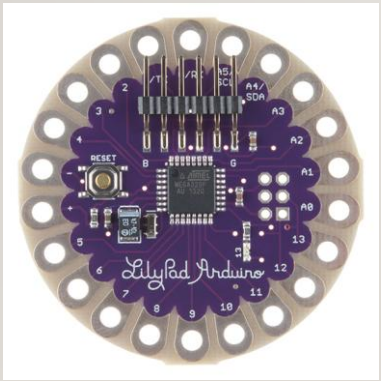
•**Fundamentos:** Se basó en el proyecto "Wiring" de Hernando Barragán (2003) y buscaba simplificar la programación de hardware para principiantes

¿QUÉ ES ARDUINO?

- Existen diferentes tarjetas basadas en microcontroladores para desarrollar sistemas embebidos, una de ellas es la tarjeta Arduino.
- Las tarjetas Arduino surgieron en el Instituto de Diseño Interactivo en Ivrea, Italia, en el año 2005.
- Fueron creadas como una herramienta de hardware libre, barata y fácil de usar para estudiantes de diseño y arte, en lugar de emplear los costosos y complejos sistemas de microcontroladores de la época.
- El proyecto buscó simplificar la programación para principiantes con conocimientos mínimos de electrónica.

¿QUÉ ES ARDUINO?





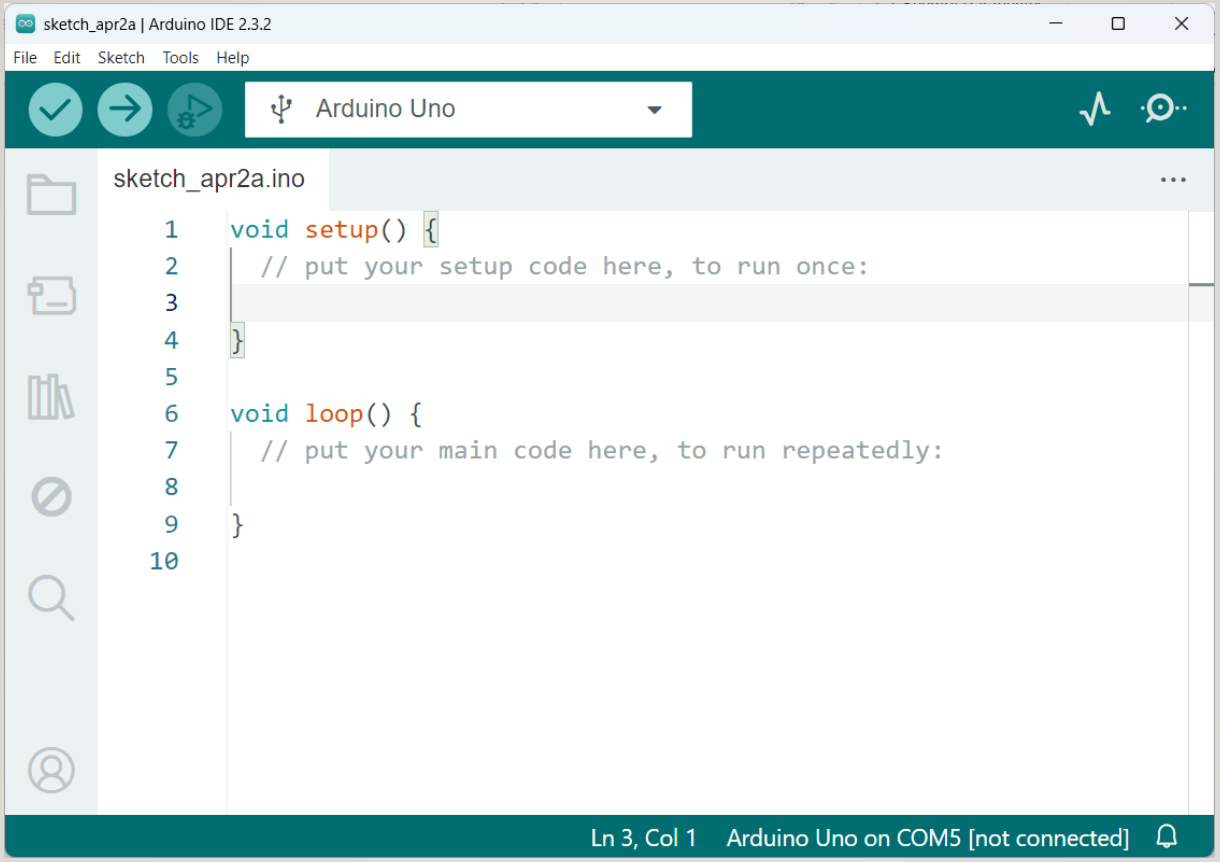
¿QUÉ ES ARDUINO?

- Además de las tarjetas, los creadores de Arduino hicieron un entorno de fácil manejo para el desarrollo de aplicaciones.
- Con Arduino no es necesario tener un conocimiento amplio del microcontrolador y tampoco se requiere de muchas habilidades en programación.
- El entorno incluye muchas bibliotecas con funciones que facilitan el manejo de periféricos, para la construcción de sistemas embebidos.

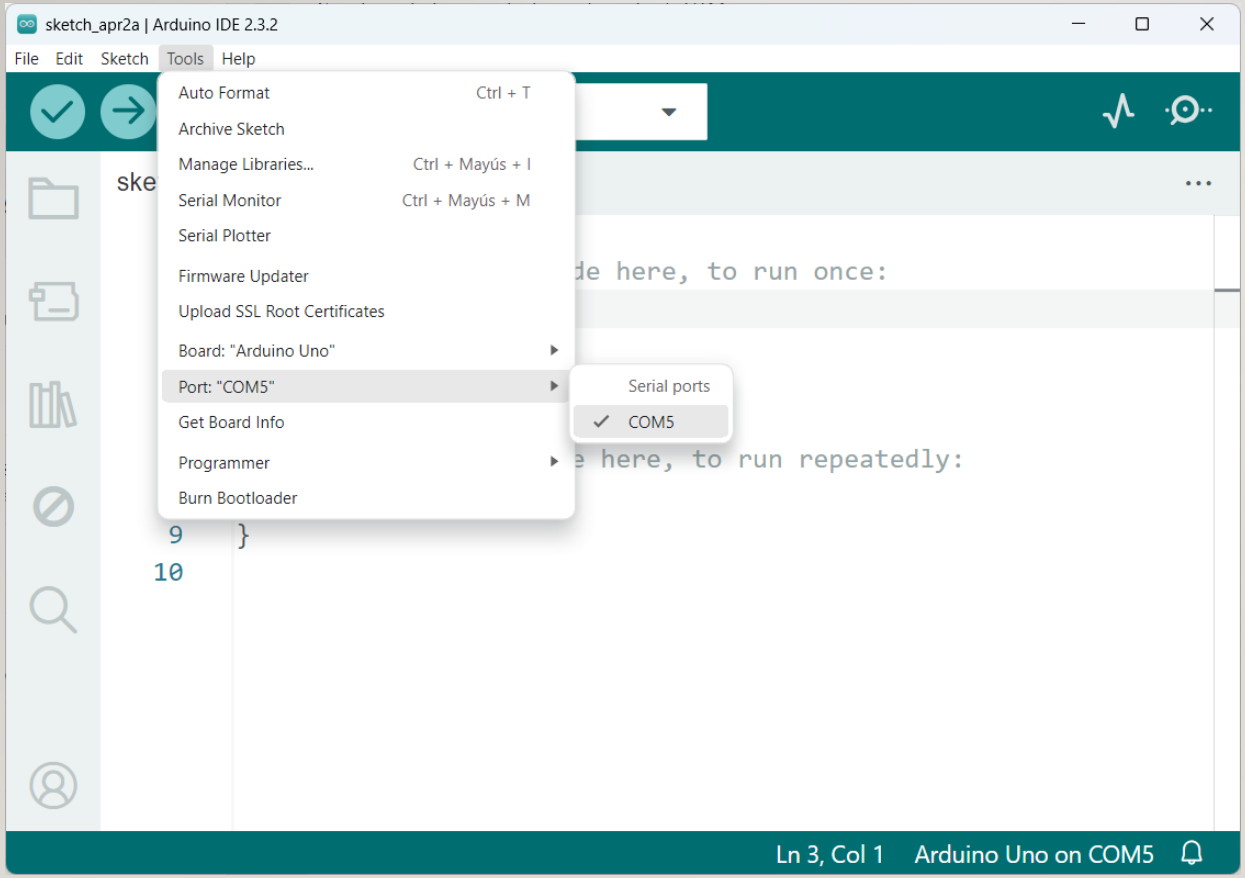
¿QUÉ ES ARDUINO?

- Arduino se realizó bajo la cultura de **hardware libre**, es decir, sus desarrolladores dejaron disponibles los diagramas de conexiones y los programas para descargar aplicaciones.
- Como consecuencia, se creó una **comunidad** muy grande que diseñó e implementó **tarjetas complementarias** o *shields* para el manejo de sensores, memorias, pantallas visuales, etc.

EL IDE DE ARDUINO

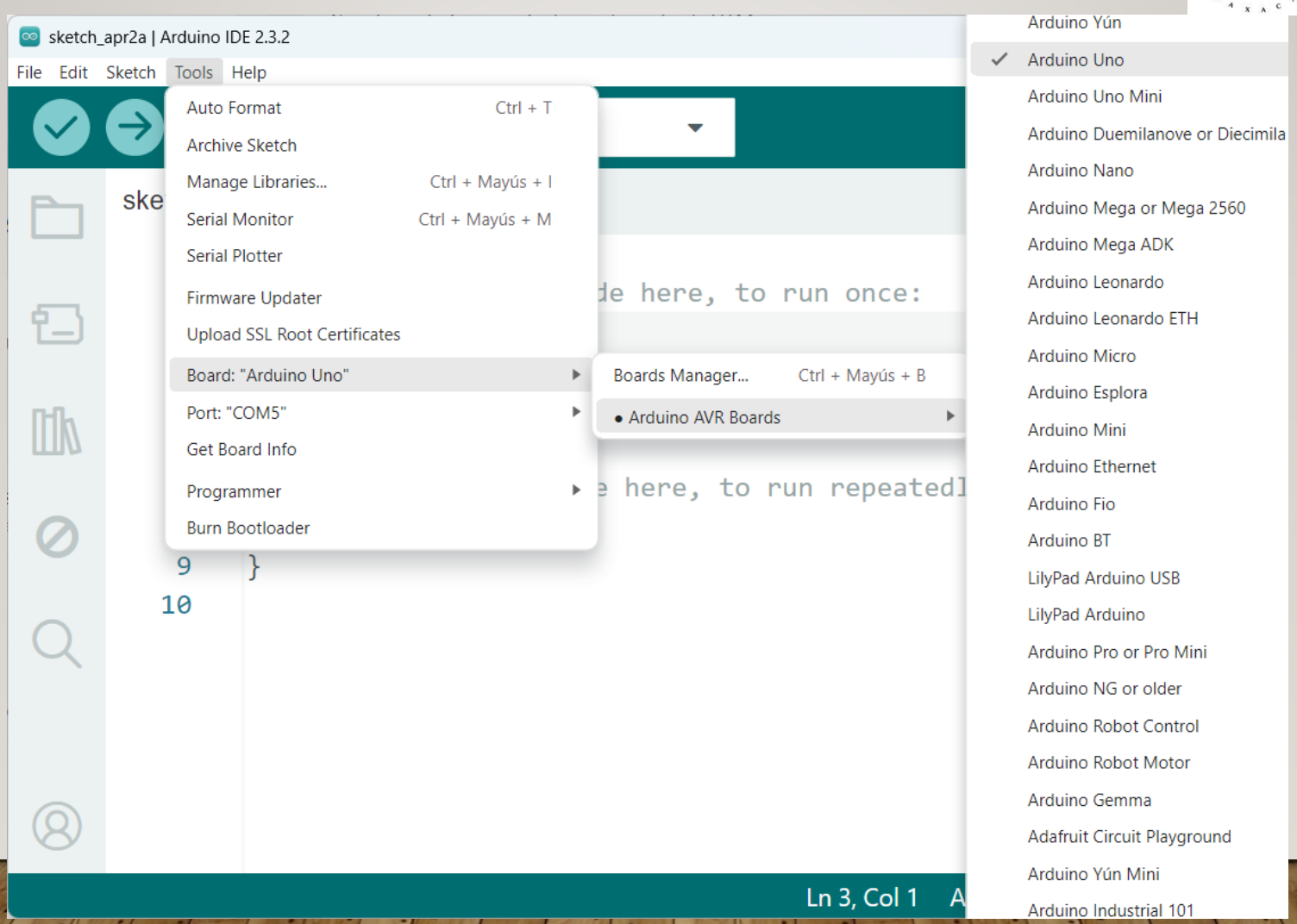


13 CONEXIÓN DE LA TARJETA

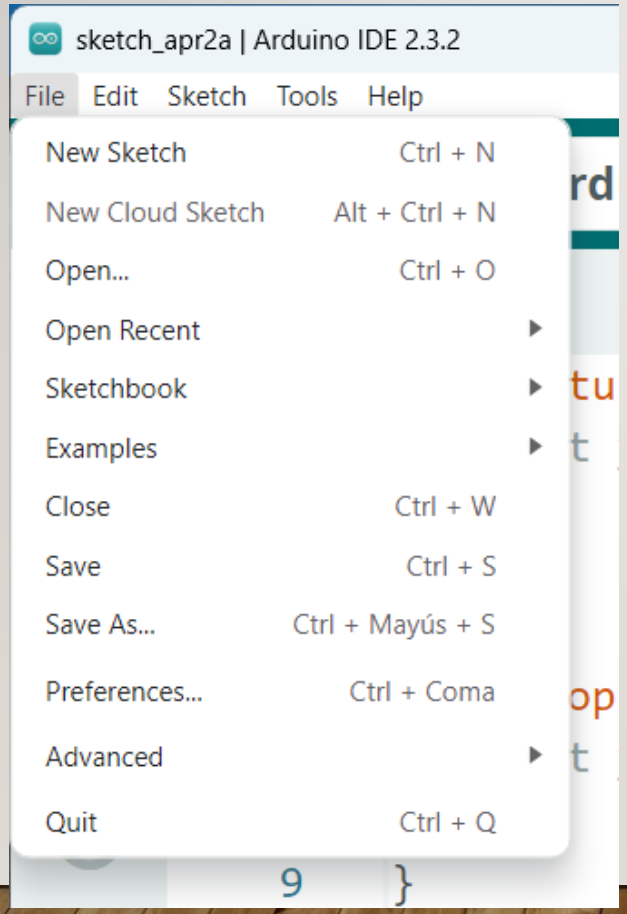


- Muestra el puerto de la computadora en donde se conecta la tarjeta.
- Si no aparece un puerto activo es porque la tarjeta no ha sido reconocida.

14 SELECCIÓN DE LA TARJETA



SKETCH



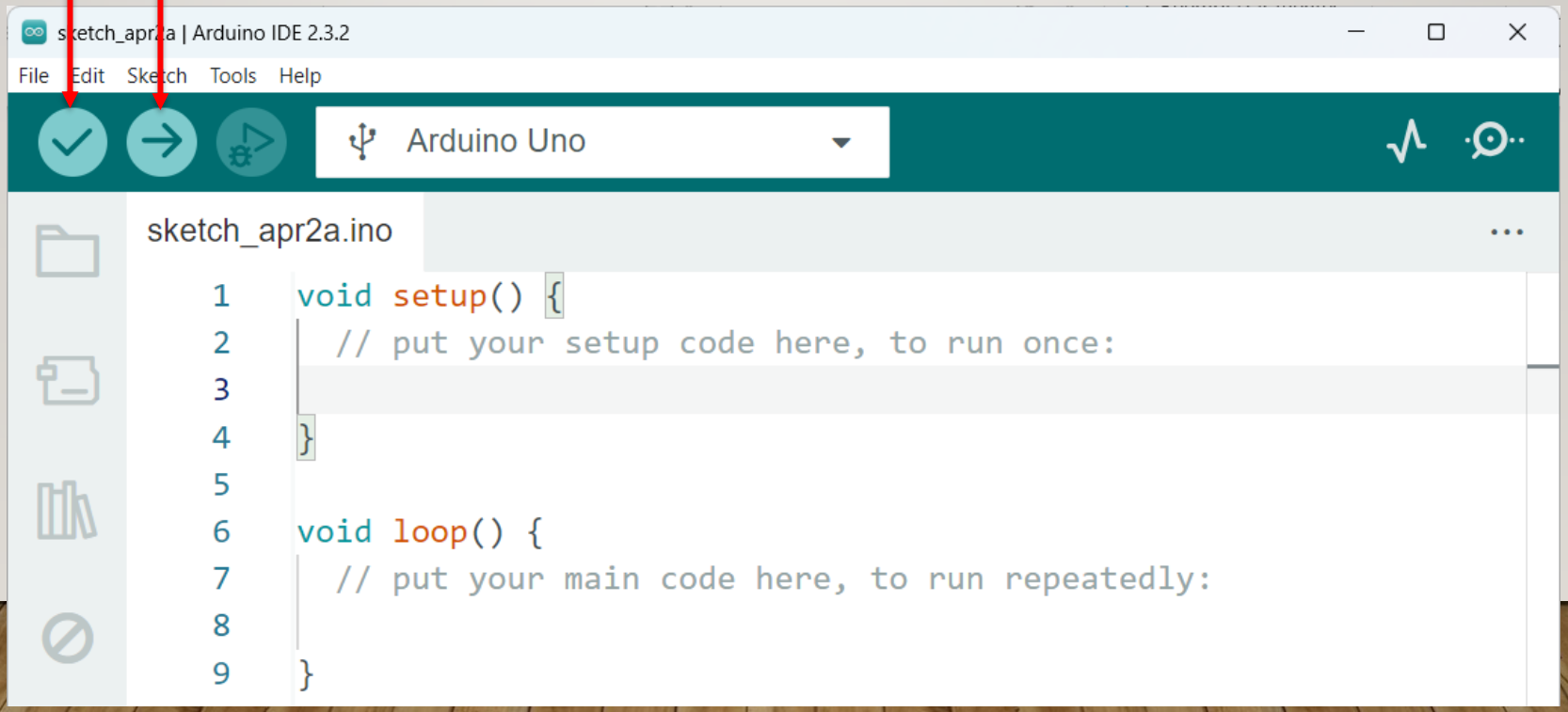
- Cada programa o archivo de Arduino se denomina **Sketch.**

- Por default, al crear un nuevo **Sketch**, este complementa su nombre con la fecha actual.

16 BOTONES DE ACCESO RÁPIDO

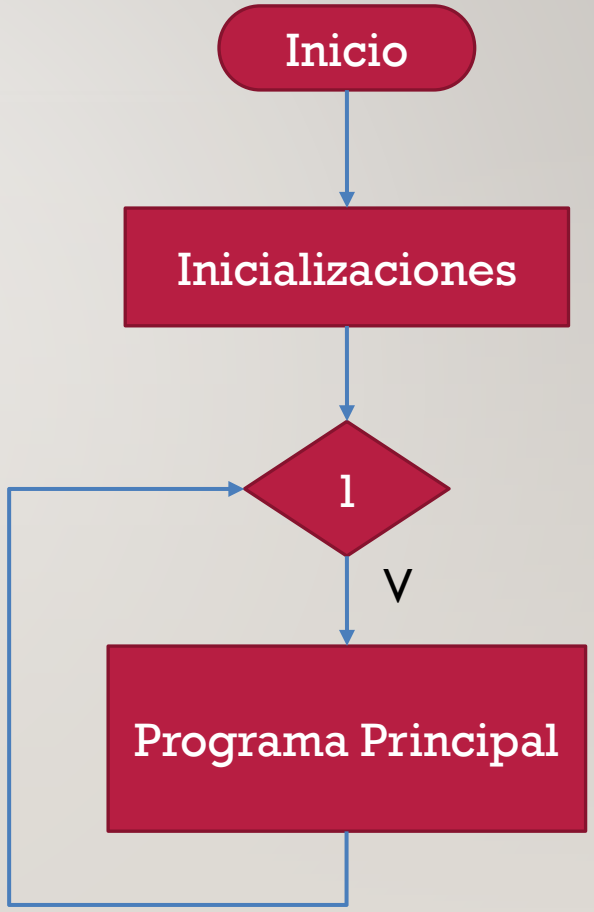
Verifica que el Sketch no tenga errores

Compila y descarga en la tarjeta

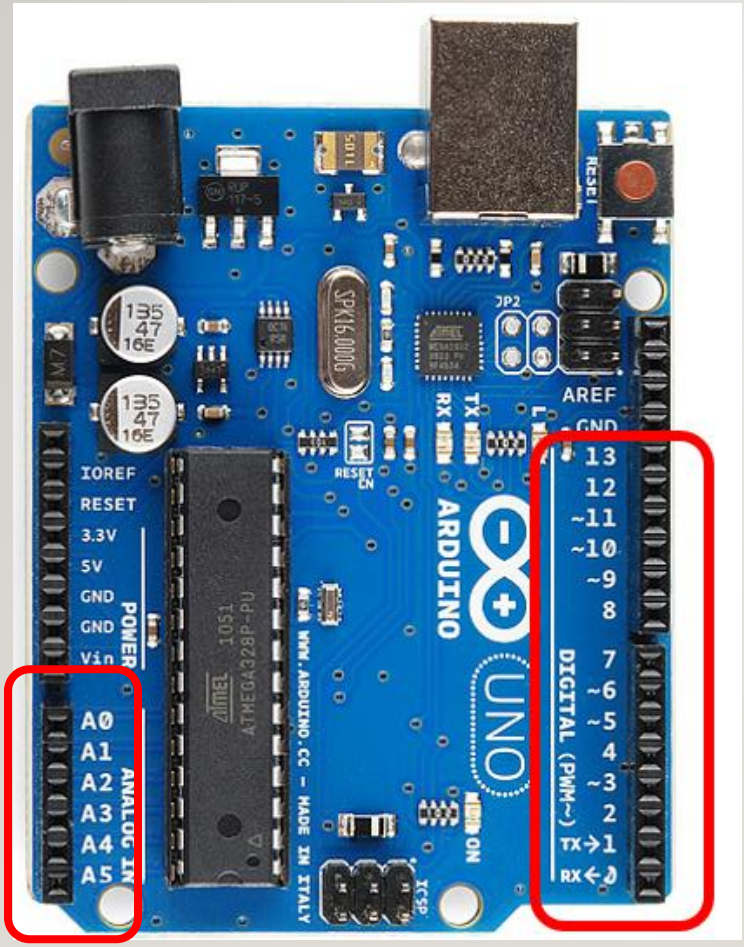


17 ORGANIZACIÓN DE UN SKETCH

```
void setup() {
  // Inicializaciones
  // Código que se ejecuta
  // una vez
}

void loop() {
  // Programa principal
  // Código que se ejecuta
  // repetidamente
}

// Se usa para los comentarios
```



TERMINALES DE LA ARDUINO UNO



La tarjeta tiene:

- 14 terminales para entradas/ salidas digitales, marcadas como 0, 1, 2, …, 13.
- 6 terminales que pueden ser entradas/salidas digitales o entradas analógicas, marcadas como A0, A1, …, A5

TERMINALES DIGITALES

Para configurar una terminal, como entrada o salida, se utiliza la función **pinMode( ).**

La función recibe 2 parámetros:

pin: El número de terminal en la tarjeta a configurar.

modo: El modo de terminal, puede ser:

INPUT : La terminal será entrada.

OUTPUT : La terminal será salida.

Ejemplo:

```
pinMode(10, OUTPUT);        // La terminal 10 será salida
```

SALIDAS DIGITALES

En una salida digital solo se puede escribir uno de dos valores:

HIGH: Nivel lógico alto.

LOW: Nivel lógico bajo.

Esto se hace mediante la función **digitalWrite( )**:

La función recibe 2 parámetros:

pin: El número de terminal en donde se va a escribir.

valor: El valor a escribir, puede ser HIGH o LOW.

Ejemplo:

```
digitalWrite(10, HIGH);       // Escribe en la terminal 10
```

FUNCIÓN PARA RETARDOS

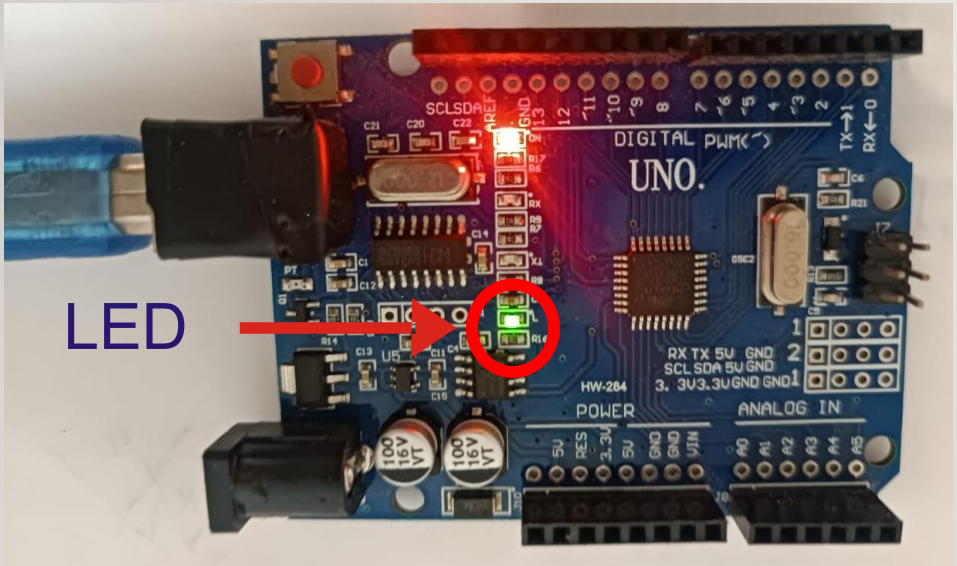
La función **delay( )** detiene la ejecución de un programa por una cantidad de milisegundos especificada como parámetro.

Ejemplos:

```
delay(500);   // Retardo de medio segundo
delay(1000);  // Retardo de un segundo
```

LED DE LA TARJETA

La tarjeta Arduino Uno tiene un LED de prueba, ubicado en la terminal 13.



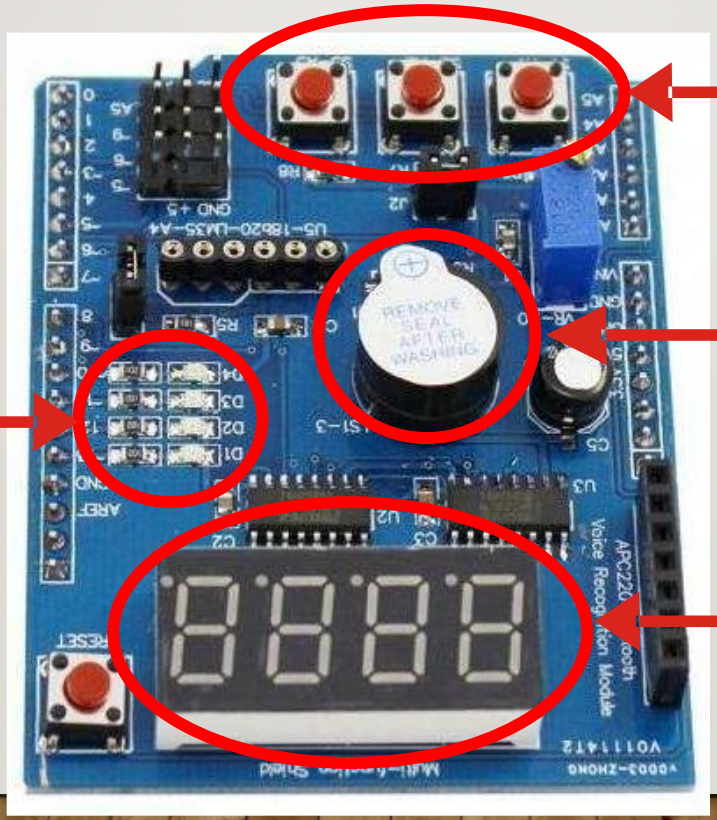
Para juntar lo revisado hasta el momento, se escribirá un Sketch que haga parpadear al LED de prueba, encendiendo al LED por medio segundo y apagándolo por medio segundo.

23 EJEMPLO 1: PARPADEO DEL LED

```
void setup() {
   // La terminal 13 debe ser salida
   pinMode(13, OUTPUT);
}

void loop() {
   // LED encendido por medio segundo
   digitalWrite(13, HIGH);
   delay(500);
   // LED apagado por medio segundo
   digitalWrite(13, LOW);
   delay(500);
}
```

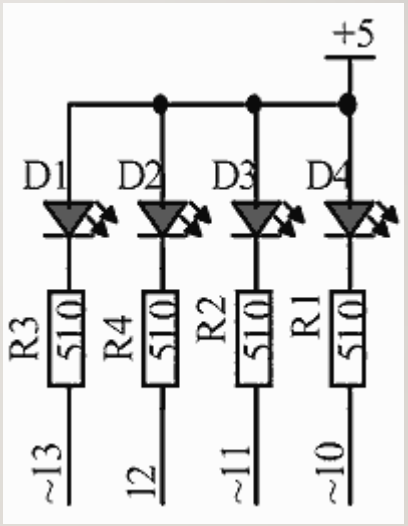
24 MANEJO DE OTROS ELEMENTOS

Para evaluar otras entradas o salidas, se empleará al *shield* multifunción.



LED'S DE LA TARJETA

- El *shield* contiene 4 LEDS, conectados en las terminales 10, 11, 12 y 13 de la tarjeta, como se muestra en la figura:



- Por la forma en que se encuentran conectados los LEDs, encienden cuando se manda un nivel lógico bajo (LOW).

EJEMPLO 2

- Realice un Sketch que alterne en el encendido de los LED's ubicados en las terminales 12 y 13.
- Es decir, cuando el LED 13 esté encendido, el LED 12 debe estar apagado, y vice-versa.
- Los cambios se deben realizar cada medio segundo.

27 OTROS ELEMENTOS DE PROGRAMACIÓN

Programas de Aplicación:

void setup() { } ….. void loop() { }

ARDUINO

Biblioteca de funciones basadas en Lenguaje C/C++

Un cargador ayuda a llevar nuevas aplicaciones a la memoria

LENGUAJE C/C++

Incluye todos los elementos del lenguaje:

Tipos de datos, operadores, estructuras de control de flujo

Microcontrolador AVR

(Tarjeta Arduino Uno, Nano o Mega)

VARIABLES

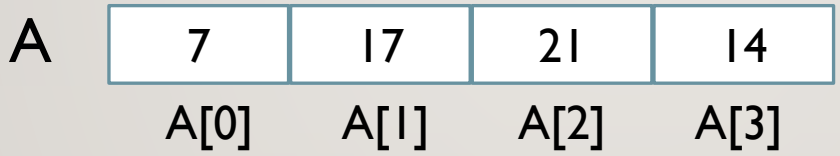
- Nombre que se le asigna a las localidades de memoria, para almacenar información acorde con su tipo.
- Las variables pueden ser del tipo entero (int) o del tipo carácter (char).
- Ejemplo de uso:

```
int  LED = 13;   // Variable entera

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

void loop() {
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}
```

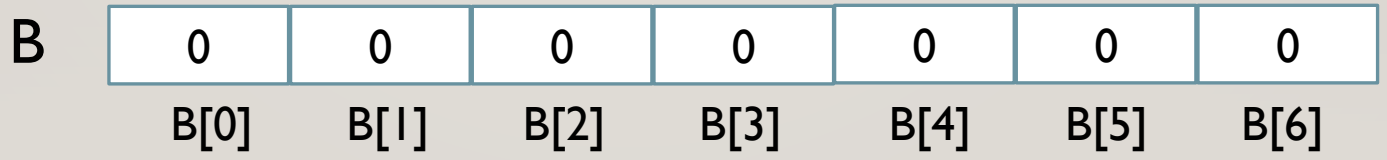
ARREGLOS

- Es una colección de variables del mismo tipo, cada elemento del arreglo se distingue por su posición.
- Ejemplo de uso:

```
int  A[] = { 7, 17, 21, 14 };   // Arreglo con 4 elementos
```




```
int  B[7];   // Arreglo con 7 elementos sin inicializar
```



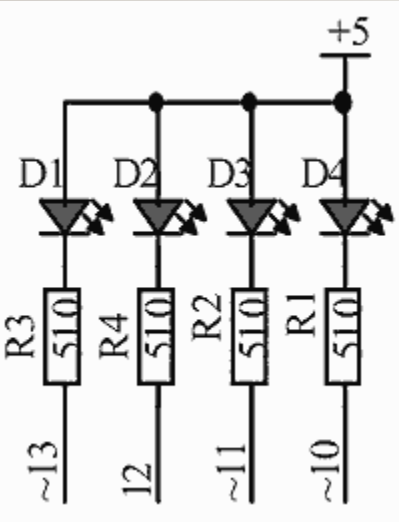
ARREGLOS

- Los arreglos se pueden consultar o modificar durante la ejecución de un programa.
- Ejemplo de uso:

```
int   x, y;

x = A[2];         // Lee al elemento 2 del arreglo (x = 21)
y = 8;
B[0] = y;        // Escribe 8 en el elemento 0 de B
B[3] = 5;        // Escribe 5 en el elemento 3 de B

int i = 2;
B[i] = 9;       // Escribe 9 en el elemento i (2) de B
```

LED'S DE LA TARJETA

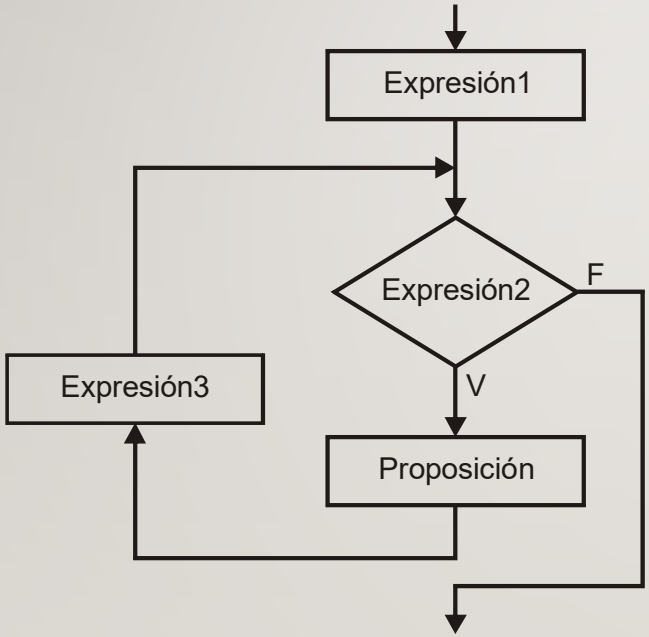
- Si todas las terminales del *shield* se van a configurar como salidas, se puede emplear un arreglo:




```
int  LEDS[] = { 13, 12, 11, 10};

void setup() {
  pinMode(LEDS[0], OUTPUT);    // D1
  pinMode(LEDS[1], OUTPUT);    // D1
  pinMode(LEDS[2], OUTPUT);    // D1
  pinMode(LEDS[3], OUTPUT);    // D1
}
```

TAREAS REPETITIVAS

El lenguaje incluye una estructura de control para tareas repetitivas:




```
// Estructura repetitiva
for (expr1; expr2; expr3) {

      proposición
}
```

La expresión 2 es una condición para repetir la proposición.

TAREAS REPETITIVAS

- La configuración de las salidas, es una tarea repetitiva:

```
int  LEDS[] = { 13, 12, 11, 10};
int  i;
void setup() {
  for( i = 0; i < 4; i++) {
    pinMode(LEDS[i], OUTPUT);
  }
}
```

```
i++ significa: i = i + 1
```

EJEMPLO 3

- Realice un Sketch que inicie con todos los LED's apagados, después encienda un LED, posteriormente 2, después 3 y finalmente 4.
- La secuencia a seguir se muestra en la tabla:

D1	D2	D3	D4
0	0	0	0
I	0	0	0
I	I	0	0
I	I	I	0
I	I	I	I

Aplique un delay de 500 ms entre cada cambio.

La secuencia se debe repetir al llegar al final.

1 - LED encendido.
0 - LED apagado.

Los LED's se encienden con LOW.

NÚMEROS BINARIOS

- El sistema decimal es posicional, los dígitos obtienen su valor por la posición:

$$325 = 3x10^2 + 2x10^1 + 5x10^0$$

- En el sistema binario es lo mismo, pero solo se tienen dos dígitos, el 0 y el 1, así, para obtener el valor decimal de un número binario se realiza la siguiente operación:

$$(1010)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 8 + 0 + 2 + 0 = 10$$

$$(1111)_2 = 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0 = 8 + 4 + 2 + 1 = 15$$

- Las computadoras almacenan la información en binario, como secuencias de 1's y 0's.

NÚMEROS BINARIOS

No.	b3	b2	b1	b0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

CONVERSIÓN DE DECIMAL A BINARIO

- Para convertir un número decimal a binario, se realizan divisiones sucesivas entre 2 y los residuos forman el resultado.
- Por ejemplo, para convertir el 13 a binario:

División	Resultado	Residuo
13/2	6	1
6/2	3	0
3/2	1	1
1/2	0	1

- Así: $13 = (1101)_2$ $(1101)_2 = 8 + 4 + 0 + 1 = 13$

EJERCICIO 4: CONTADOR BINARIO

- Genere un contador de 0 a 15, con salida en código binario.
- La salida se mostrará en los 4 LEDS del *shield* multi-función.
- Aplique un delay de 500 ms entre cada cambio.
- Después del número 15, el contador debe reiniciar en 0.

No.	D4	D3	D2	D1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

CONTADOR BINARIO

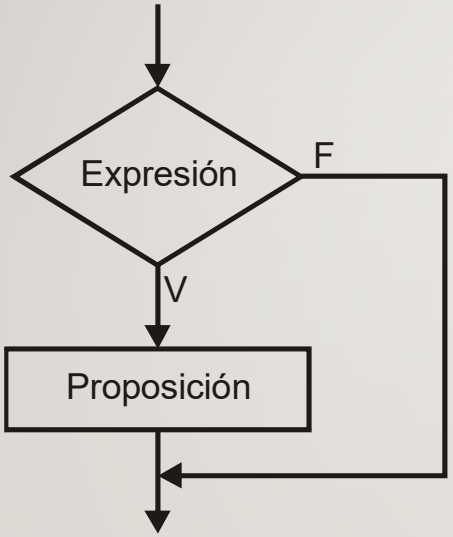
ANÁLISIS:

- Se llevará un **contador decimal** que se **convertirá a binario**.
- La conversión implica 4 divisiones sucesivas, el LED se enciende solo **si el residuo es 1** (se debe tomar una decisión).
- En lenguaje C el residuo se obtiene con el **operador %**.

No.	D4	D3	D2	D1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

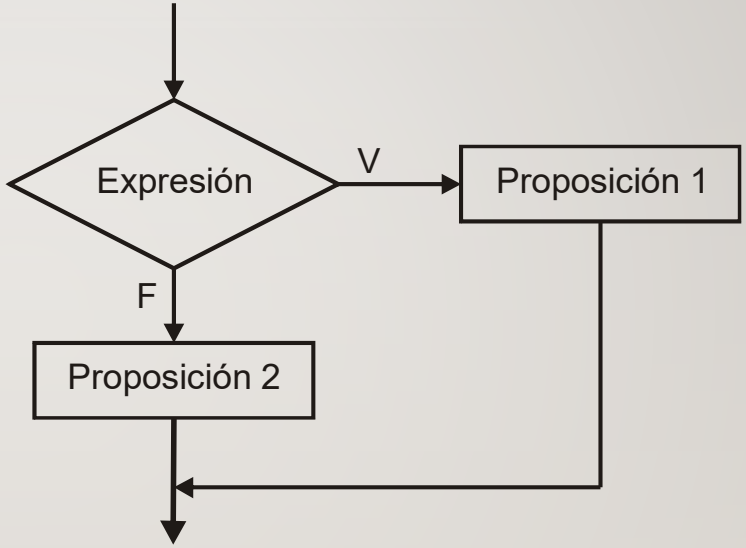
ESTRUCTURAS DE DECISIÓN

Decisión Simple




```
if(expresión)
   Proposición
```

Decisión Doble




```
if(expresión)
   proposición1
else
   proposición2
```

41 CONVERSIÓN DE DECIMAL A BINARIO

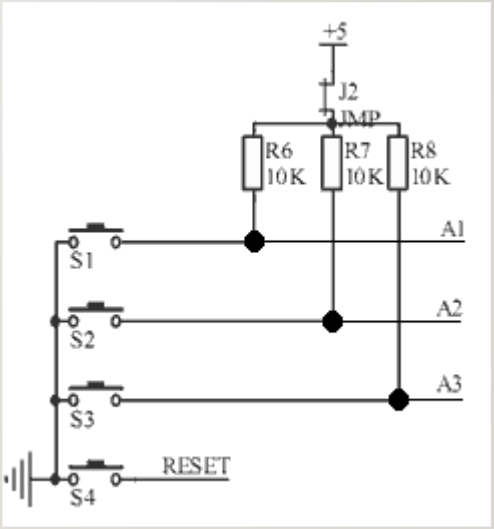
- Se realizan 4 divisiones, se utiliza una variable auxiliar para no perder el valor del contador:

```
aux = conta;
for( i = 0; i < 4; i++) {
    if( aux % 2 == 1) {                 // Residuo en 1
        digitalWrite(LEDS[i], LOW);     // Enciende el LED
    }
    else {                              // Residuo en 0
        digitalWrite(LEDS[i], HIGH);    // Apaga al LED
    }
    aux = aux/2;                        // Actualiza
}
```

Se utiliza == en una comparación e = en una asignación.

BOTONES DE ENTRADA

- El *shield* contiene 3 botones conectados de en las terminales A1, A2 y A3, como se muestra en la figura:



- En las terminals se lee un nivel alto (HIGH) cuando el botón no está presionado y un nivel lógico bajo (LOW) cuando se presiona.

ENTRADAS DIGITALES

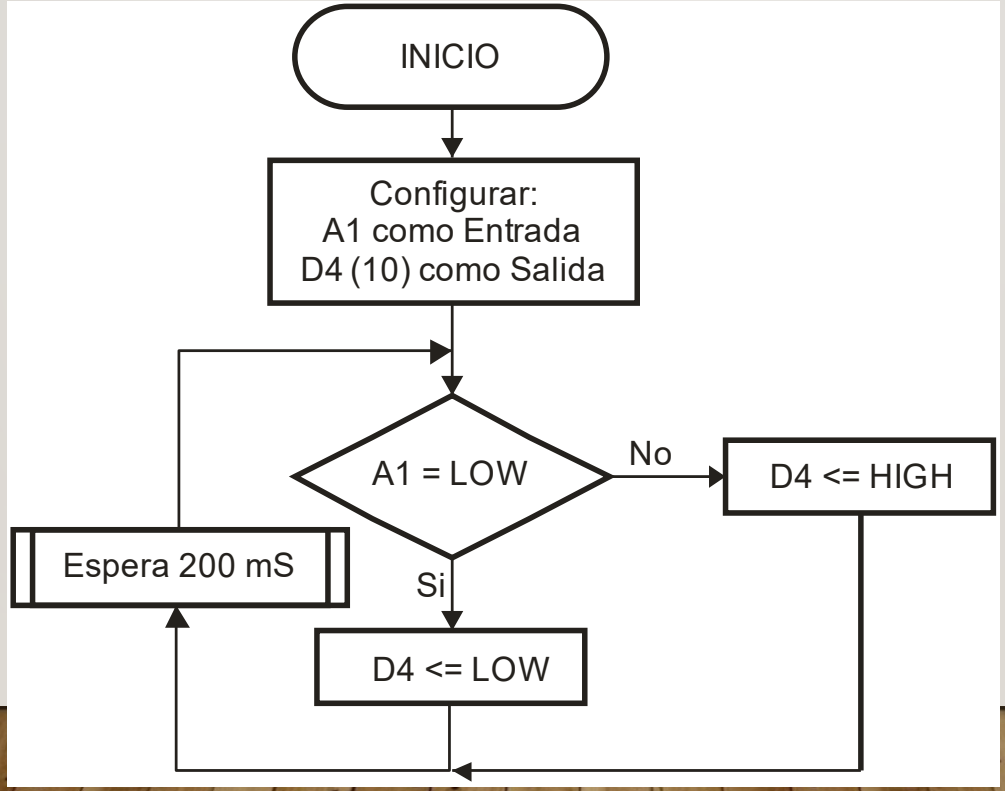
Las entradas digitales se leen con la función **digitalRead**( ), la función recibe el número de la terminal a leer y regresa HIGH o LOW.

Ejemplo:

```
int val;                        // Para el valor leido
val = digitalRead(A1);          // Lectura de un botón
if( val == LOW ) {
   // Acciones cuando el botón se presiona
}
```

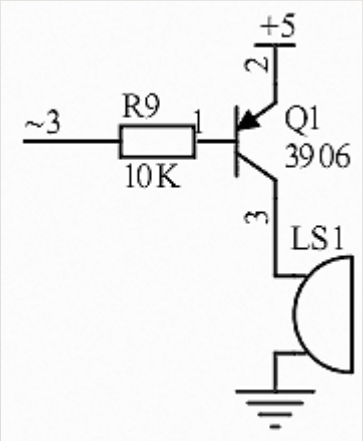
EJERCICIO 5

Refleje el estado del botón conectado en A1 en el LED ubicado en D4 (terminal 10), si el botón está presionado se debe encender el LED, en caso contrario debe estar apagado.



EJERCICIO 6
EVALUACIÓN DEL BUZZER

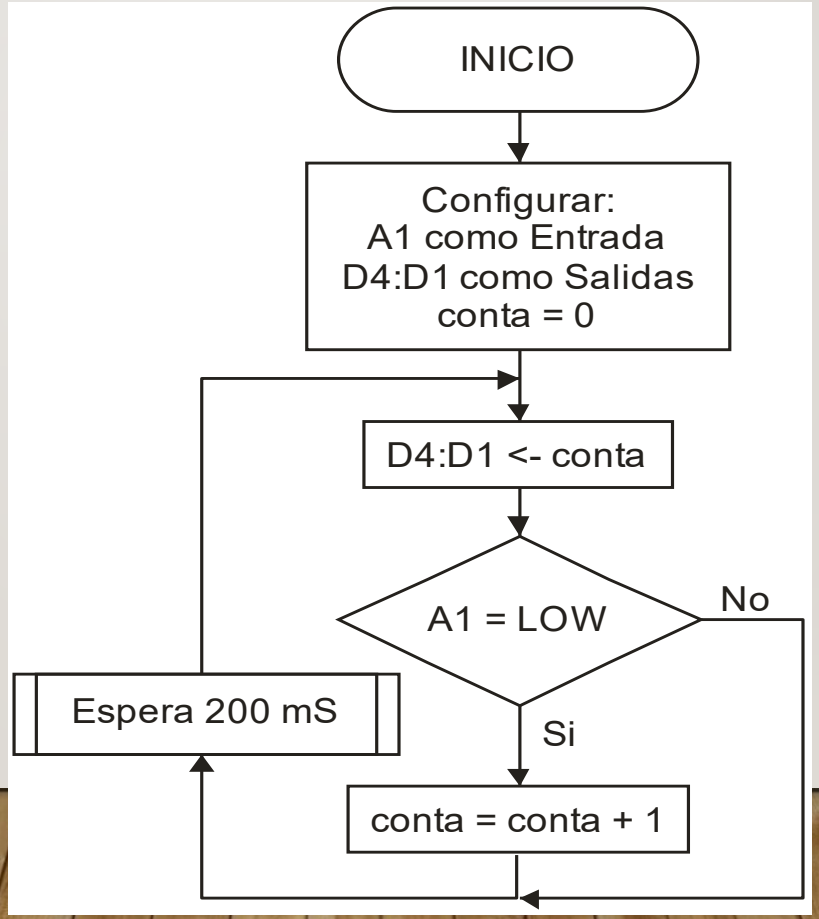
El *shield* multifunción tiene un *buzzer* (zumbador) conectado en la salida digital número 3, el cual se activa con un nivel lógico bajo, como se muestra en la figura:



Modifique el ejercicio anterior para el *buzzer* se active cuando se presione el botón A1.

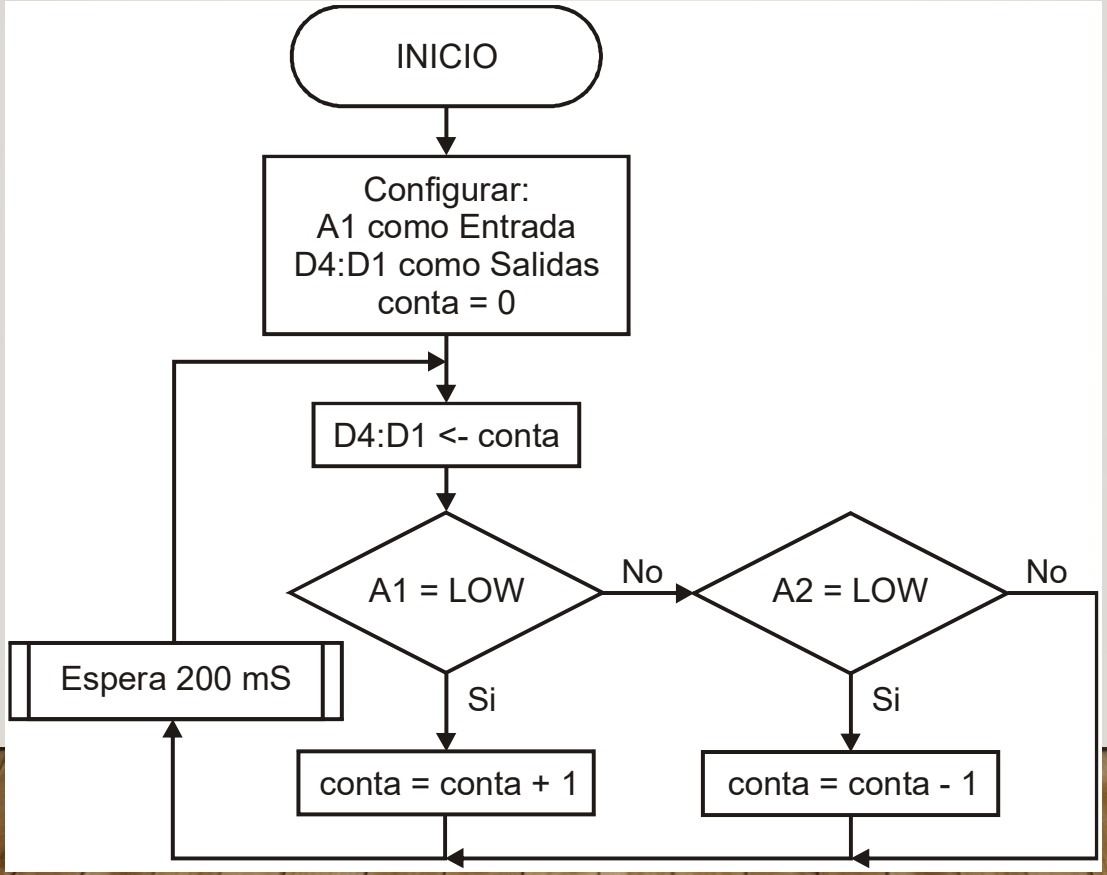
EJERCICIO 7
CONTADOR BINARIO DE EVENTOS

Modifique el contador desarrollado en el ejercicio 4, de manera que la salida se incremente cuando se presione el botón conectado en la terminal A1.



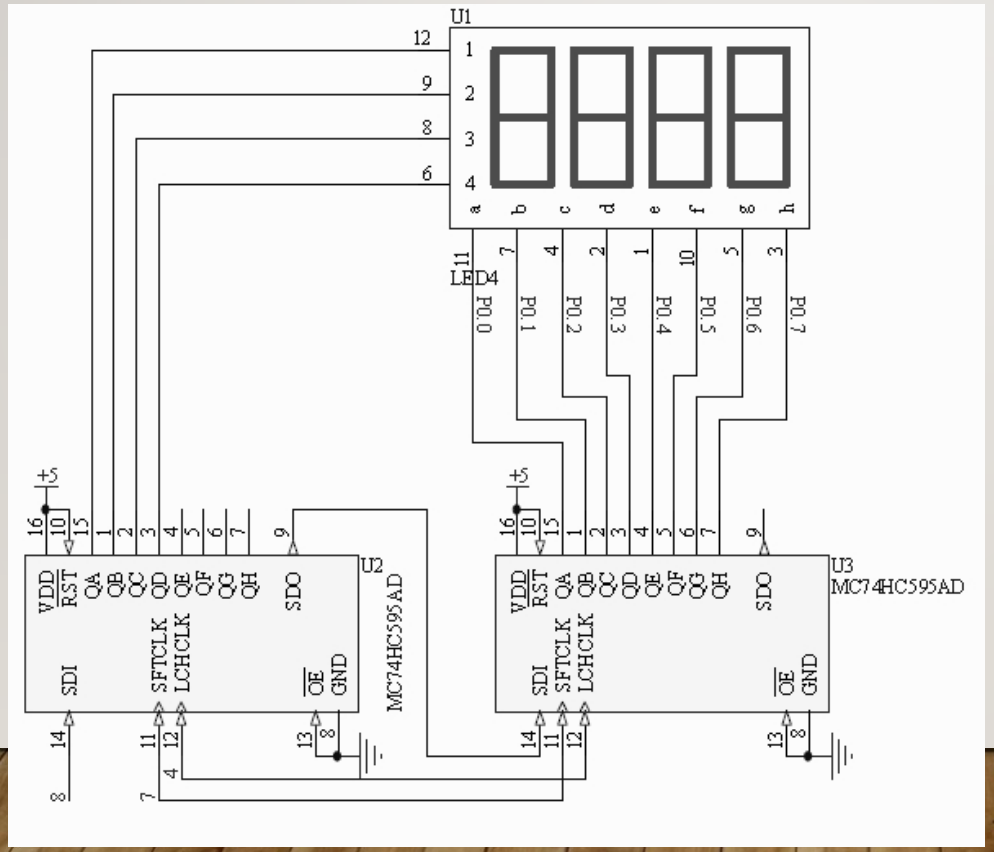
EJERCICIO 8
CONTADOR UP/DOWN

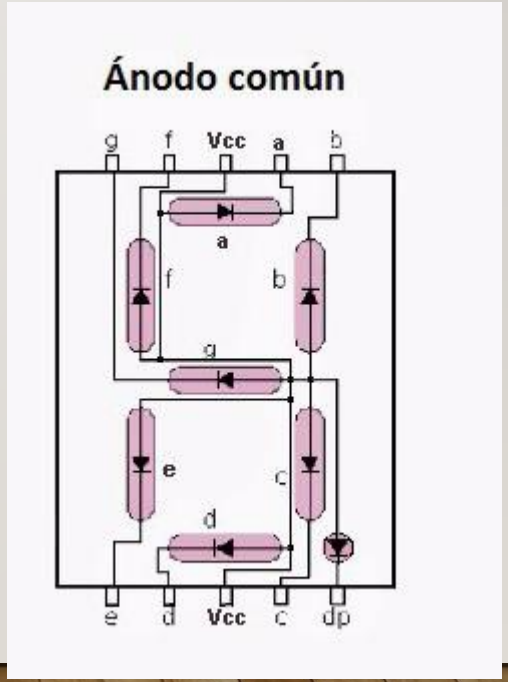
Modifique el contador desarrollado en el ejercicio anterior considerando otro botón, la salida se incrementará cuando se presione el botón A1 y decrementará si se presiona el botón A2.



DISPLAYS DE 7 SEGMENTOS

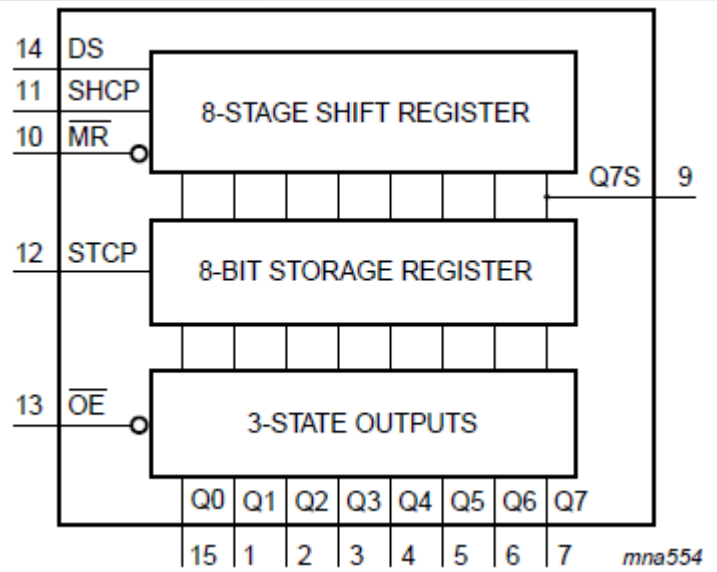
- El *shield* multifunción tiene 4 displays de 7 segmentos de ánodo común, conectados mediante un bus para los datos y habilitadores independientes.





49 DISPLAYS DE 7 SEGMENTOS

- Los displays son manejados con dos circuitos integrados de la serie MC74HC595AD, que corresponde a un registro de desplazamiento para realizar una conversión serie a paralelo, con un espacio de almacenamiento y salida en tercer estado.



El chip tiene relojes independientes para el desplazamiento y para el almacenamiento.

Así como una terminal de salida para una conexión en cascada.

DISPLAYS DE 7 SEGMENTOS

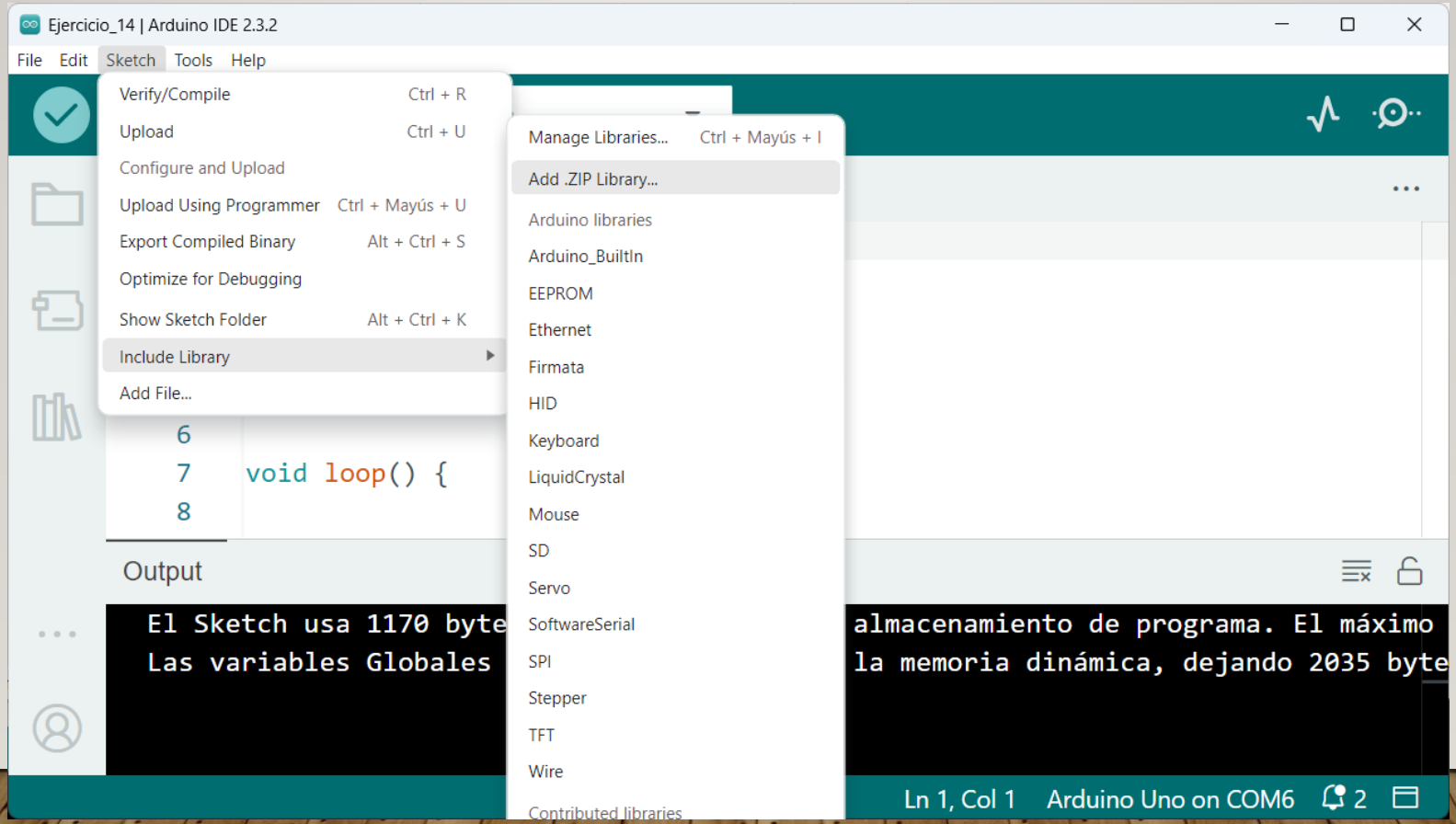
- El manejo de los Display de 7 segmentos es una tarea compleja, ya que para mostrar un número en un display, se debe enviar la información bit a bit, sincronizando con una señal de reloj.

- Para facilitar las tareas complejas, el entorno de Arduino permite agregar bibliotecas desarrolladas por terceros, incrementando su funcionalidad.

- Para el manejo de los displays del *shield* multifunción, se utilizará la biblioteca Displays_MFS.

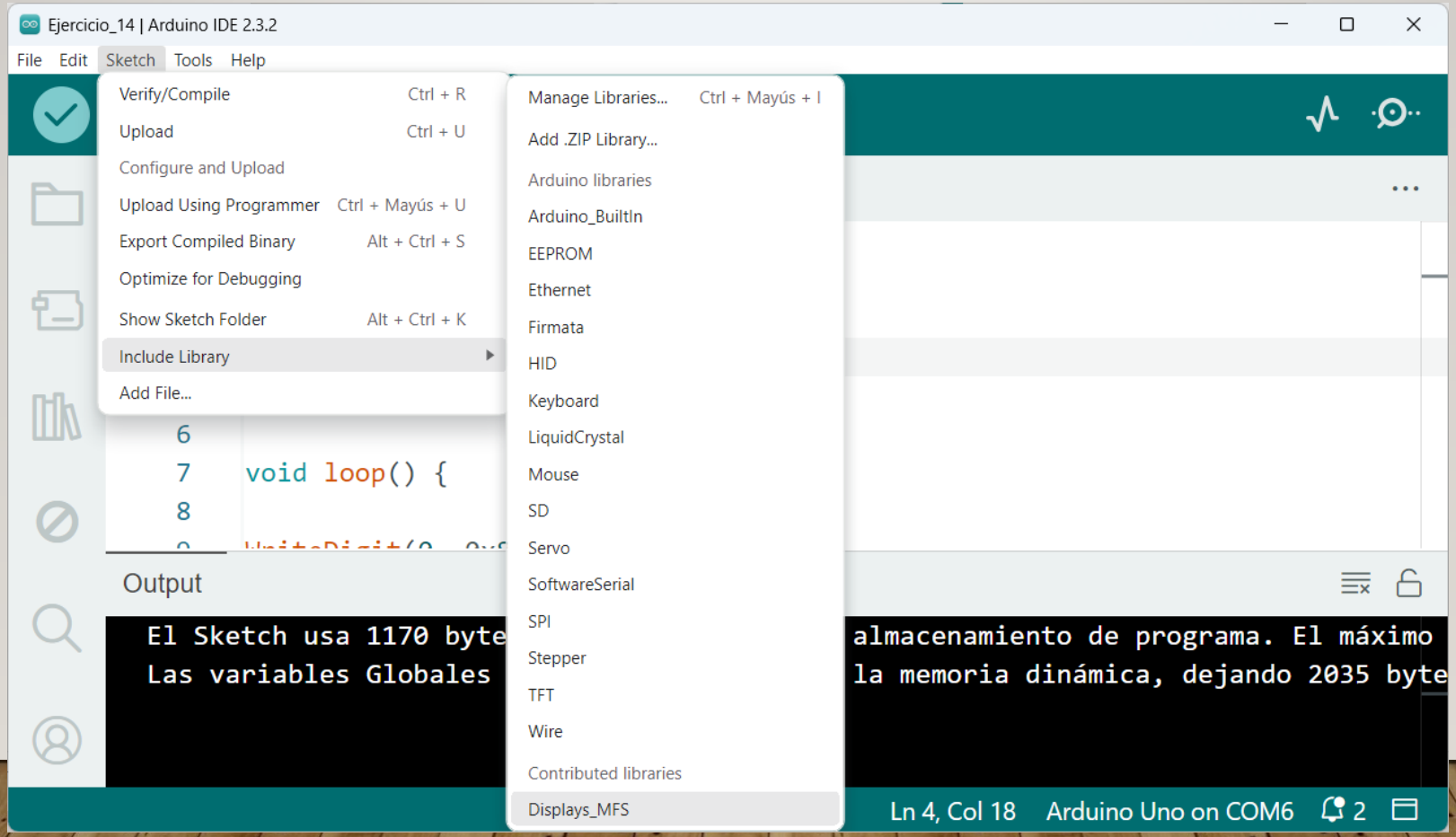
51 DISPLAYS DE 7 SEGMENTOS

- La biblioteca se agrega como un archivo ZIP.



DISPLAYS DE 7 SEGMENTOS

- El entorno debe reconocer la nueva biblioteca.



DISPLAYS DE 7 SEGMENTOS

La biblioteca Display_MFS incluye las funciones:

- **initDisplays**() : Configura las terminales para el manejo de los displays, deben ser salidas.
- **WriteDigit**(*Segment*, *Value*) : Escribe el dato recibido en *Value*, en el display indicado por *Segment*. *Segment* debe ser un número entre 0 y 3, *Value* hace referencia a los 7 segmentos más el punto decimal.
- **WriteInteger**(*Data*) : Escribe un entero entre -999 y 9999, fuera de ese rango escribe una indicación de error. La función hace un recorrido en los 4 displays, se debe llamar continuamente para mantener la información.

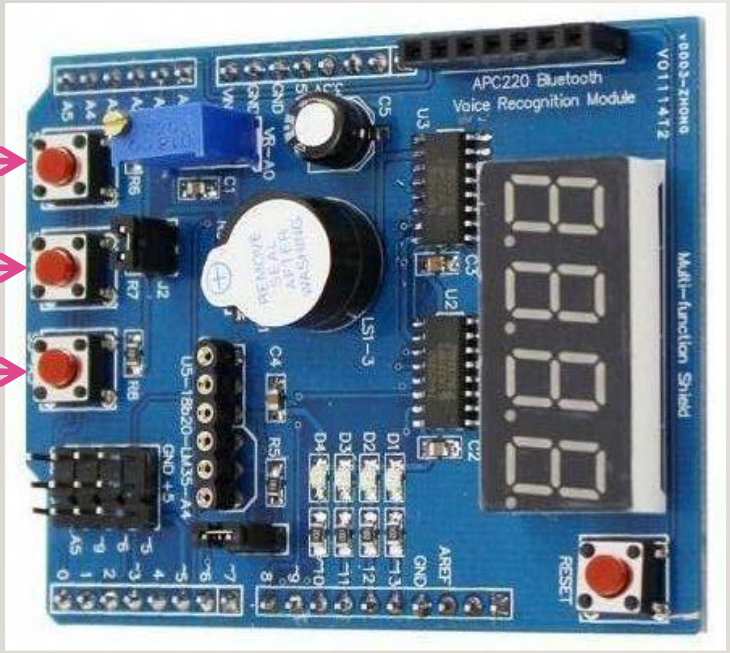
EJERCICIO 9

- Evalúe la funcionalidad de la biblioteca con este Sketch.
- **0b** indica que el número es binario, los LED's se encienden con 0 y se apagan con 1.
- El de la derecha corresponde con el segmento A y el de la izquierda con el punto decimal.
- Pruebe lo que sucede cuando el delay se reduce a 100 y 5 ms.

```
#include <Displays_MFS.h>

void setup() {
  initDisplays();
}

void loop() {
  WriteDigit(0, 0b11111001); // 1
  delay(500);
  WriteDigit(1, 0b10100100); // 2
  delay(500);
  WriteDigit(2, 0b10110000); // 3
  delay(500);
  WriteDigit(3, 0b10011001); // 4
  delay(500);
}
```

EJERCICIO 10
CONTADOR UP-DOWN

Realice un contador de eventos con salida en los 4 displays de 7 segmentos, del *shield* multifunción se utilizarán los 3 botones como se muestra en la figura:



- Utilice la función **WriteInteger**(*Data*), debe llamar contimuamente a la función.
- No utilice delay's, en lugar de ello, haga 10 llamadas a la función **WriteInteger**().

EJERCICIO 11
CARACTERES PERSONALIZADOS

Utilice la función **WriteDigit** de la biblioteca Display_MFS.h para mostrar la cadena constante HOLA, como se muestra la figura:

Letra	dp	g	f	e	d	c	b	a	HEX	DEC
H	1	0	0	0	1	0	0	1	0x89	137
O	1	1	0	0	0	0	0	0	0xC0	192
L	1	1	0	0	0	1	1	1	0xC7	199
A	1	0	0	0	1	0	0	0	0x88	136

GRACIAS POR SU ATENCIÓN

M. C. Felipe Santiago Espinosa

fsantiag@mixteco.utm.mx
felipe.santiago.e@gmail.com

marzo de 2026