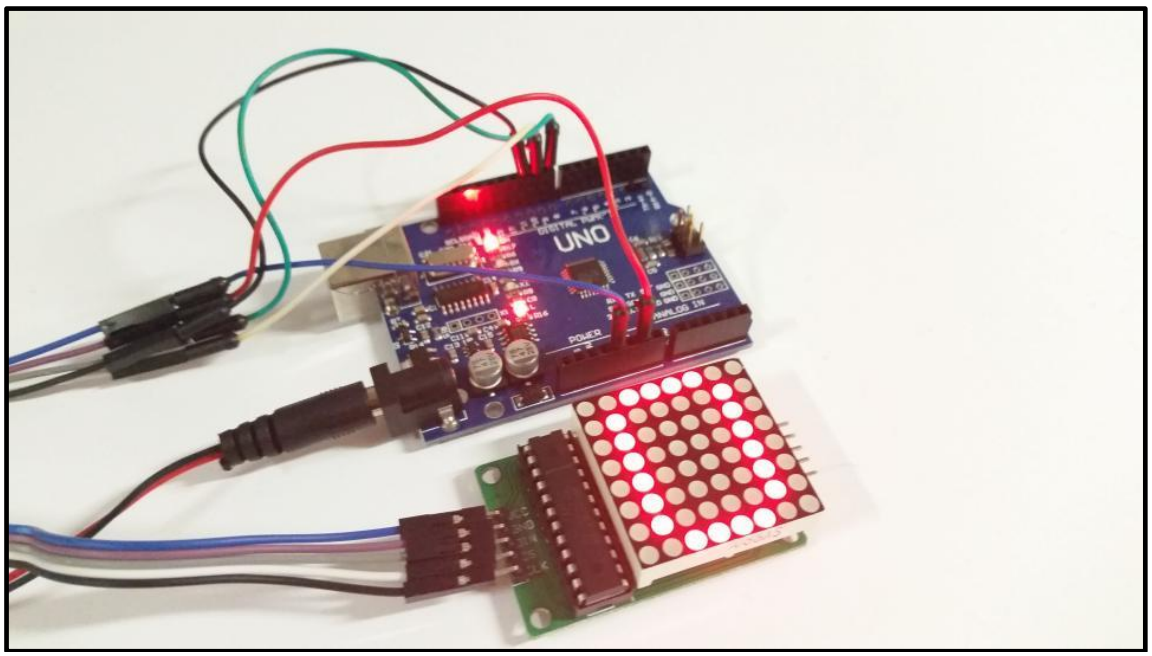
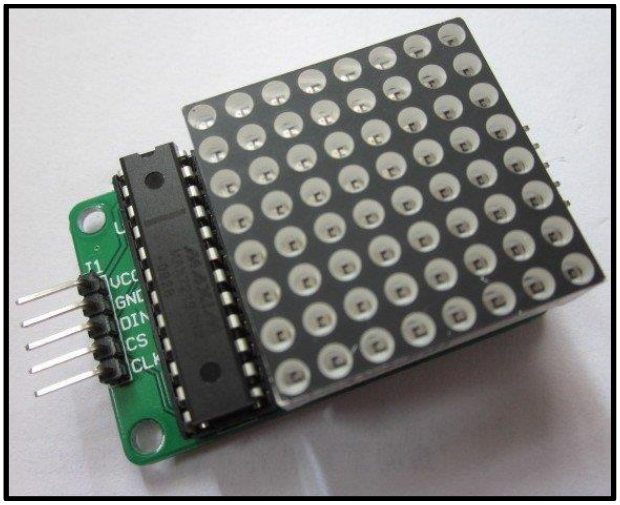
Display de 8x8 LEDs controlado con el integrado MAX 7219 y Arduino



Registro de desplazamiento MAX 7219

El controlador de display MAX7219 es un circuito integrado de registro de desplazamiento de 16 bits. Los primeros 8 bits definen un comando, mientras que los segundos 8 bits contienen los datos para ese comando.

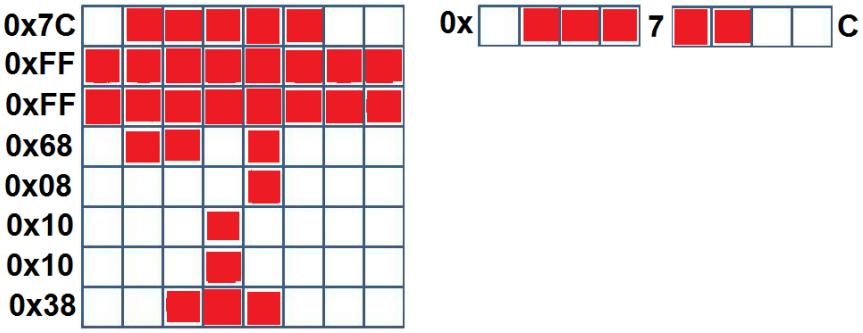
De esta forma, utilizando tres pines de salida digital de Arduino se pueden controlar los 64 LEDs de un display de 8X8 LEDs. El conjunto de registro de desplazamiento y display es barato y se puede comprar por menos de 5 €.



El display dispone de cinco terminales de conexión. Los terminales VCC y GND se utilizan para la alimentación eléctrica a 5 voltios. En los siguientes ejemplos el terminal CLK está conectado al pin 10 de Arduino, el terminal CS al pin 9 y el terminal DIN al pin 8.

Modo de codificar los diferentes LEDs que se han de encender

Vamos a crear un programa que genere el siguiente dibujo en el display.



El display tiene 8 líneas de ocho LEDs cada una. Cada línea se codifica con cuatro dígitos y los dos primeros son siempre “0x”. El tercer dígito indica, en hexadecimal, el valor de los primeros cuatro LEDs empezando por la izquierda y el cuarto dígito el valor de los siguientes cuatro LEDs.

Para el caso de una letra A mayúscula se puede ver cuál sería la codificación en la imagen siguiente.

	C1				C2				Binario		Hexadecimal
L1					X				0000	1000	08
L2				X		X			0001	0100	14
L3			X				X		0010	0010	22
L4			X	X	X	X	X		0011	1110	3E
L5			X				X		0010	0010	22
L6			X				X		0010	0010	22
L7			X				X		0010	0010	22
L8			X				X		0010	0010	22

En la tabla siguiente se pueden ver las equivalencias entre números binarios y hexadecimales.

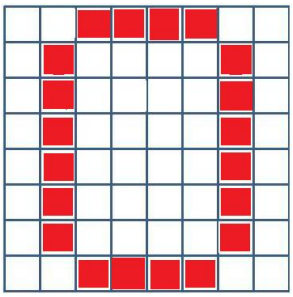
DECIMAL	BINARIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

De esta forma, la siguiente línea de código, que muestra el cero dibujado en el display,

{0x3C,0x42,0x42,0x42,0x42,0x42,0x42,0x3C},//0

Indica, línea a línea, los LEDs que estarán encendidos.

La primera línea es la correspondiente a 0x3C, la parte significativa es 3C. Esto nos indica que en la primera parte de la matriz tendremos un 3 (0011) y en la segunda parte una C (1100), todo junto quedará como 00111100. Esta será la disposición de encendido de los LEDs de la primera fila de la matriz. De la misma forma se codifican las 7 líneas siguientes.

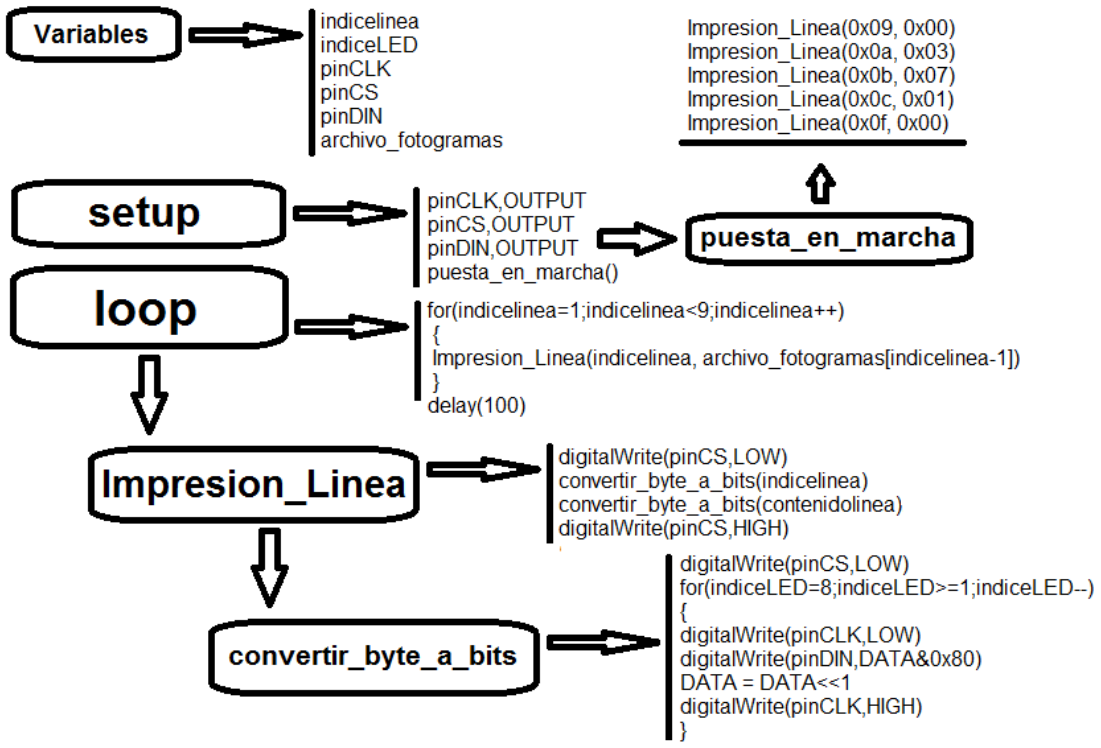


Volviendo al caso del dibujo del principio, el programa completo para Arduino es el siguiente.

```
int indicelinea;
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[8]=

{0x7C,0xFF,0xFF,0x68,0x08,0x10,0x10,0x38};
  void convertir_byte_a_bits(char DATA)
  {
      digitalWrite(pinCS,LOW);
         for(indiceLED=8;indiceLED>=1;indiceLED--)
           {
            digitalWrite(pinCLK,LOW);
            digitalWrite(pinDIN,DATA&0x80);       //Extrae cada bit del byte "DATA"
            DATA = DATA<<1;
            digitalWrite(pinCLK,HIGH);
           }
  }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
       digitalWrite(pinCS,LOW);
       convertir_byte_a_bits(indicelinea);     //Envia el valor de la linea del display
       convertir_byte_a_bits(contenidolinea); //Envia el valor de los LEDs de la
                                                 // linea del display
       digitalWrite(pinCS,HIGH);
}
```

```
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01); //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT
                                    //del display :0
}

void setup()
{
  pinMode(pinCLK,OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(pinDIN,OUTPUT);
  delay(50);
  puesta_en_marcha();
}

void loop()
{
      for(indicelinea=1;indicelinea<9;indicelinea++)
       {
       Impresion_Linea(indicelinea, archivo_fotogramas[indicelinea-1]);
       }
      delay(100); //Tiempo de espera entre fotograma y fotograma
}
```

Este es el diagrama de bloques del programa.



La primera parte del programa es la declaración de variables. Cinco de ellas son de tipo “int” (indicelinea, indiceLED, pinCLK, pinCS y pinDIN. La variable archivo_fotogramas es una matriz de tipo “char”.

Después de las variables se definen cinco funciones. Dos de ellas (setup y loop) son comunes a cualquier programa de Arduino. Las otras tres funciones son puesta_en_marcha, Impresion_Linea y convertir_byte_a_bits.

Desde la función “setup” se llama a la función “puesta_en_marcha”. Desde la función “loop” se llama a la función “Impresion_Linea” y desde la función “Impresion_Linea” se llama a la función “convertir_byte_a_bits”, que es la que finalmente enciende y apaga los LEDs de cada línea del display.

Ejercicio 1

Ahora haz un programa similar al anterior, pero con el dibujo de un coche.

Generación de una animación con varios fotogramas

Basándose en el programa anterior se puede conseguir una animación, imprimiendo sucesivos fotogramas en el display.

El siguiente programa produce una animación que nos muestra un “bitman” que primero baila y después corre hacia la derecha.

Para ello necesitamos modificar la matriz de datos:

```
char archivo_fotogramas[8]=
{0xFF,0x7C,0x3C,0x5A,0x5A,0x24,0x24,0x66};
```

de forma que sea una matriz de doble entrada y permita guardar muchos fotogramas, tantos como queramos. Por tanto esta matriz quedará definida de esta forma:

```
char archivo_fotogramas[41][8]={

{0x18,0x18,0x3C,0x5A,0x5A,0x24,0x24,0x66},//0
{0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//1
{0x00,0x30,0x3C,0xDA,0x3C,0x28,0x68,0x0C},//2 ……..
```

De los 41 fotogramas, el que tiene el número de orden “0” es el primero

```
{0x18,0x18,0x3C,0x5A,0x5A,0x24,0x24,0x66},//0
```

Y dentro de este el tercer bloque (La tercera línea del display) es el

```
archivo_fotogramas[0][2] = 0x3C
```

De hecho si escribimos esto en Arduino:

archivo_fotogramas[0][2] = 0x3C;

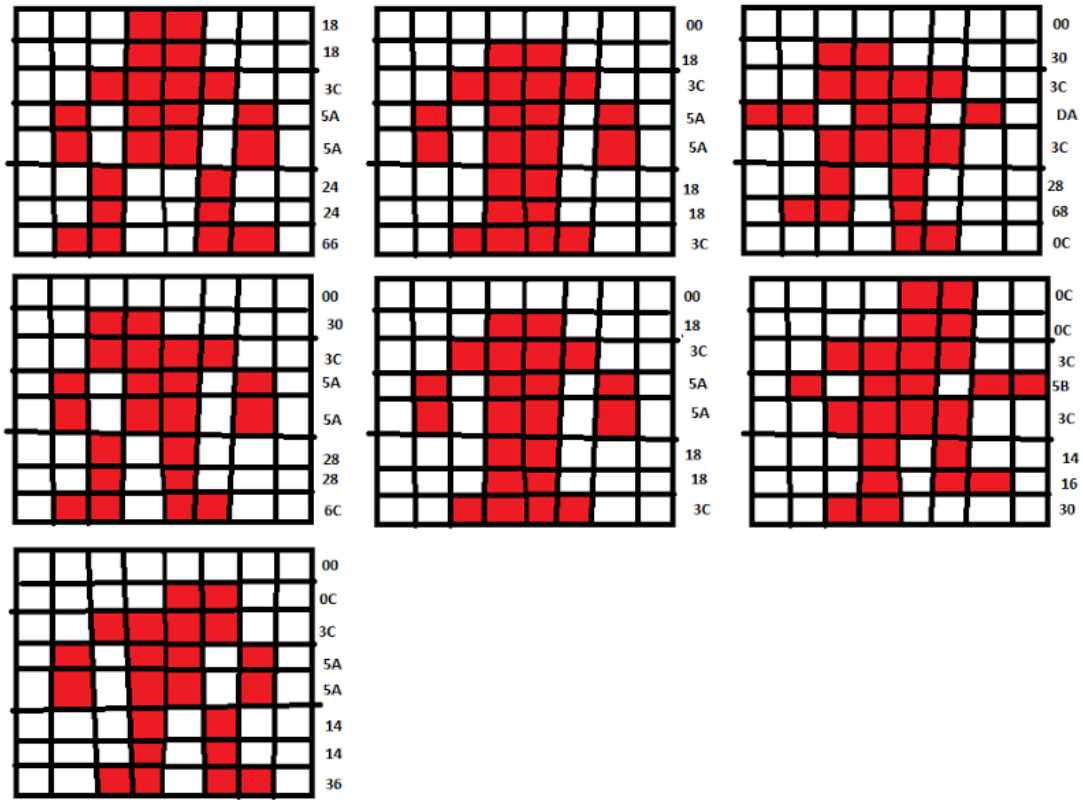
estamos asignando el valor “0x3C” a la variable “archivo_fotogramas[0][2]”.

En el “loop” hemos de hacer dos bucles para ir imprimiendo cada una de las 8 líneas de cada uno de los 41 fotogramas.

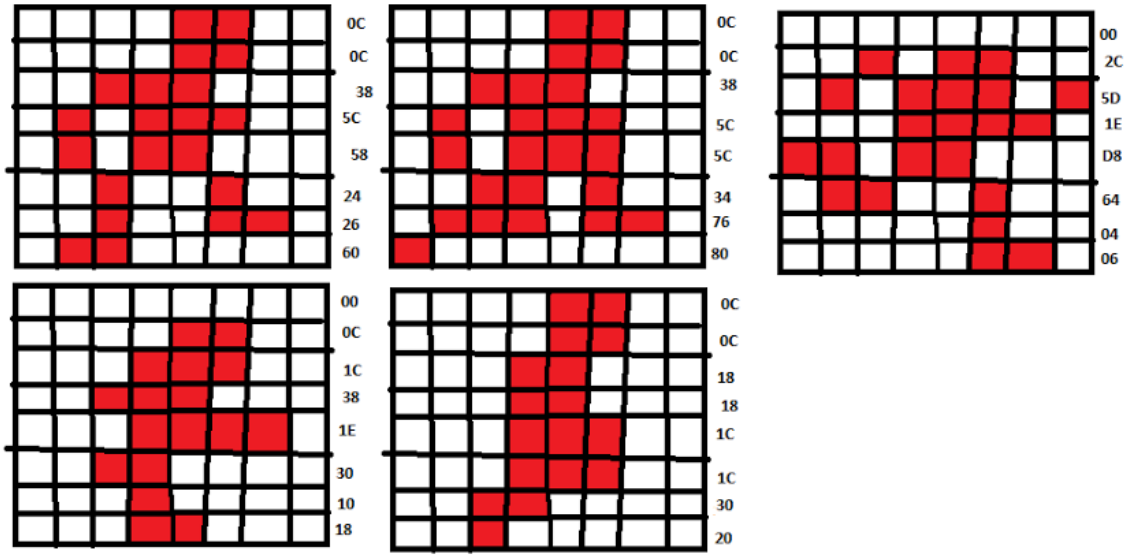
```
for(indicefotograma=0;indicefotograma<41;indicefotograma++)
   {
     for(indicelinea=1;indicelinea<9;indicelinea++)
      {
       Impresion_Linea(indicelinea,
archivo_fotogramas[indicefotograma][indicelinea-1]);
      }
     delay(100); //Tiempo de espera entre fotograma y fotograma
   }
```

El primer bucle va variando el valor de la variable “índicefotograma” y dentro de este un segundo bucle hace variar la variable “indicelinea” de 1 a 8, para cada uno de los valores de la variable “indicefotograma”.

Puestos a ello, estos son los fotogramas de la secuencia del baile con su código hexadecimal al lado.



A continuación se pueden ver los fotogramas de la carrera, con su codificación en hexadecimal.



Una vez definidos todos los fotogramas, a continuación se muestra el programa completo.

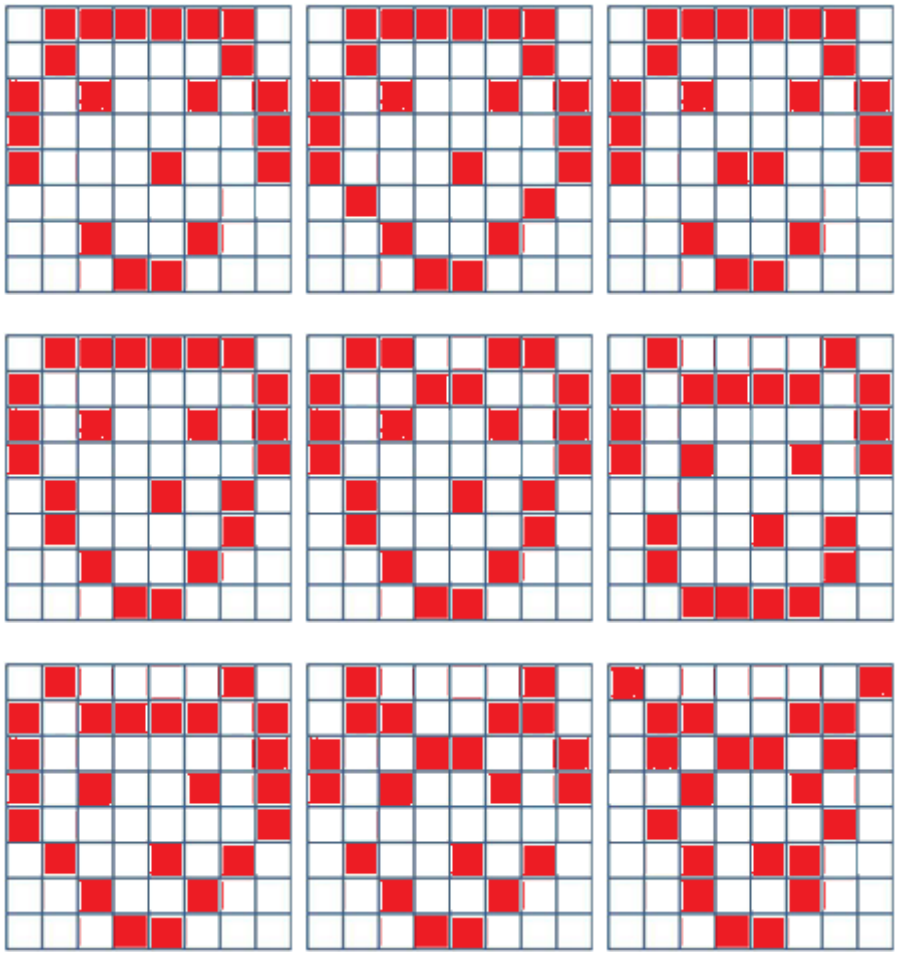
```
int indicefotograma;
int indicelinea;
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[41][8]={

        {0x18,0x18,0x3C,0x5A,0x5A,0x24,0x24,0x66},//0
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//1
        {0x00,0x30,0x3C,0xDA,0x3C,0x28,0x68,0x0C},//2
        {0x00,0x30,0x3C,0x5A,0x5A,0x28,0x28,0x6C},//3
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//4
        {0x0C,0x0C,0x3C,0x5B,0x3C,0x14,0x16,0x30},//5
        {0x00,0x0C,0x3C,0x5A,0x5A,0x14,0x14,0x36},//6
        {0x18,0x18,0x3C,0x5A,0x5A,0x24,0x24,0x66},//7
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//8
        {0x00,0x30,0x3C,0xDA,0x3C,0x28,0x68,0x0C},//9
        {0x00,0x30,0x3C,0x5A,0x5A,0x28,0x28,0x6C},//10
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//11
        {0x0C,0x0C,0x3C,0x5B,0x3C,0x14,0x16,0x30},//12
        {0x00,0x0C,0x3C,0x5A,0x5A,0x14,0x14,0x36},//13
        {0x18,0x18,0x3C,0x5A,0x5A,0x24,0x24,0x66},//14
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//15
        {0x00,0x30,0x3C,0xDA,0x3C,0x28,0x68,0x0C},//16
        {0x00,0x30,0x3C,0x5A,0x5A,0x28,0x28,0x6C},//17
        {0x00,0x18,0x3C,0x5A,0x5A,0x18,0x18,0x3C},//18
        {0x0C,0x0C,0x3C,0x5B,0x3C,0x14,0x16,0x30},//19
        {0x00,0x0C,0x3C,0x5A,0x5A,0x14,0x14,0x36},//20
        {0x0C,0x0C,0x38,0x5C,0x58,0x24,0x26,0x60},//21
        {0x0C,0x0C,0x38,0x5C,0x5C,0x34,0x76,0x80},//22
```

```
          {0x00,0x2C,0x5D,0x1E,0xD8,0x64,0x04,0x06},//23
          {0x00,0x0C,0x1C,0x38,0x1E,0x30,0x10,0x18},//24
          {0x0C,0x0C,0x18,0x18,0x1C,0x1C,0x30,0x20},//25
          {0x0C,0x0C,0x38,0x5C,0x58,0x24,0x26,0x60},//26
          {0x0C,0x0C,0x38,0x5C,0x5C,0x34,0x76,0x80},//27
          {0x00,0x2C,0x5D,0x1E,0xD8,0x64,0x04,0x06},//28
          {0x00,0x0C,0x1C,0x38,0x1E,0x30,0x10,0x18},//29
          {0x0C,0x0C,0x18,0x18,0x1C,0x1C,0x30,0x20},//30
          {0x0C,0x0C,0x38,0x5C,0x58,0x24,0x26,0x60},//31
          {0x0C,0x0C,0x38,0x5C,0x5C,0x34,0x76,0x80},//32
          {0x00,0x2C,0x5D,0x1E,0xD8,0x64,0x04,0x06},//33
          {0x00,0x0C,0x1C,0x38,0x1E,0x30,0x10,0x18},//34
          {0x0C,0x0C,0x18,0x18,0x1C,0x1C,0x30,0x20},//35
          {0x0C,0x0C,0x38,0x5C,0x58,0x24,0x26,0x60},//36
          {0x0C,0x0C,0x38,0x5C,0x5C,0x34,0x76,0x80},//37
          {0x00,0x2C,0x5D,0x1E,0xD8,0x64,0x04,0x06},//38
          {0x00,0x0C,0x1C,0x38,0x1E,0x30,0x10,0x18},//39
          {0x0C,0x0C,0x18,0x18,0x1C,0x1C,0x30,0x20},//40
};

  void convertir_byte_a_bits(char DATA)
  {
     digitalWrite(pinCS,LOW);
        for(indiceLED=8;indiceLED>=1;indiceLED--)
          {
           digitalWrite(pinCLK,LOW);
           digitalWrite(pinDIN,DATA&0x80);            //Extrae cada bit del byte "DATA"
           DATA = DATA<<1;
           digitalWrite(pinCLK,HIGH);
          }
  }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
       digitalWrite(pinCS,LOW);
       convertir_byte_a_bits(indicelinea);          //Envia el valor de la linea del display
       convertir_byte_a_bits(contenidolinea);     //Envia el valor de los LEDs de la linea
                                                  //del display
       digitalWrite(pinCS,HIGH);
}

void puesta_en_marcha()
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01);       //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT del
                                    //display :0
}

void setup()
        {
          pinMode(pinCLK,OUTPUT);
          pinMode(pinCS,OUTPUT);
```

```
            pinMode(pinDIN,OUTPUT);
            delay(50);
            puesta_en_marcha();
          }

void loop()
{
  for(indicefotograma=0;indicefotograma<41;indicefotograma++)
   {
     for(indicelinea=1;indicelinea<9;indicelinea++)
      {
      Impresion_Linea(indicelinea, archivo_fotogramas[indicefotograma][indicelinea-1]);
      }
     delay(100); //Tiempo de espera entre fotograma y fotograma
   }
}
```

Ejercicio 2

Ahora has de crear un programa que genere una animación con estos fotogramas y otros que crees tú mismo.



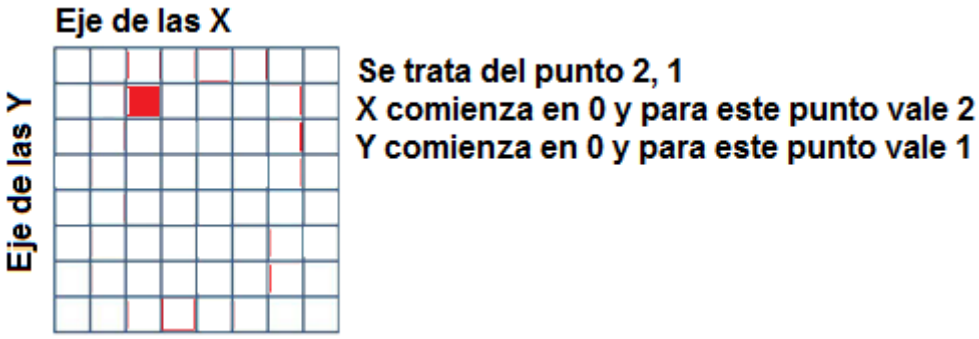
Ahora vamos a mover un punto luminoso por la pantalla

En este montaje disponemos además del display, de dos potenciómetros conectados a los pines A0 y A1 y de dos pulsadores conectados en los pines digitales 6 y 7.

El siguiente programa nos permite chequear las entradas de los dos potenciómetros.

```
int x;
int y;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  x = analogRead(A1);
  y = analogRead(A0);
  Serial.print("Valor de x:");
  Serial.println(x);
  Serial.print("Valor de y:");
  Serial.println(y);
  delay(1000);
}
```

Con los valores de estas entradas analógicas modificaremos los valores de las coordenadas del punto luminoso que queremos mover por la pantalla.

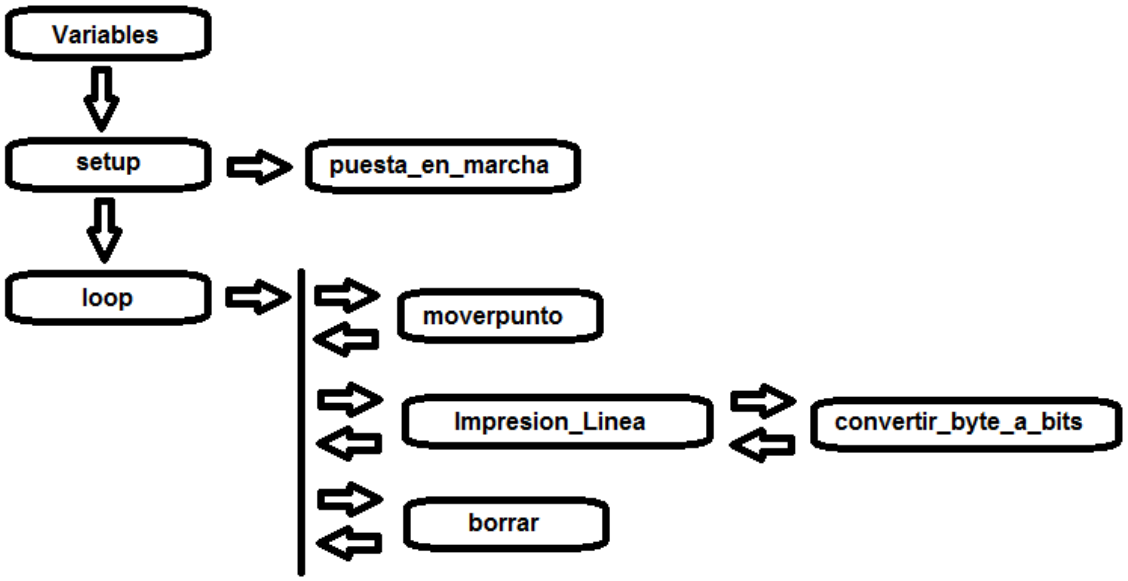


Primeramente ajustamos los valores de “x” e “y” mediante el comando map.

```
int x;
int y;

void setup()
{
```

```
  Serial.begin(9600);
}

void loop()
{
  x = analogRead(A1);
  x = map(x,0,1023,0,7);
  y = analogRead(A0);
  y = map(y,0,1023,0,7);
  Serial.print("Valor de x:");
  Serial.println(x);
  Serial.print("Valor de y:");
  Serial.println(y);
  delay(3000);
}
```

A continuación se muestra el diagrama de bloques del programa. Vemos que dispone de dos funciones nuevas, una de ellas es “moverpunto” y la otra “borrar”. A estas dos funciones se accede desde “loop”.



La función “moverpunto” calcula primeramente las coordenadas del punto, en función de los valores de entrada de los potenciómetros conectados a A0 y A1. A continuación “dibuja” el punto en la variable correspondiente, tal como se muestra a continuación:

```
if (x==0)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x80;
  }

  if (x==1)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x40;
```

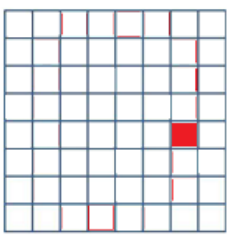
```
}

if (x==2)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x20;
}

if (x==3)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x10;
}

if (x==4)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x08;
}

if (x==5)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x04;
}

if (x==6)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x02;
}

if (x==7)
{
  archivo_fotogramas[y] = archivo_fotogramas[y] + 0x01;
}
```

Dependiendo del valor de X (De 0 a 7) asigna a “archivo_fotogramas[y]”, es decir, a la línea en que se encuentre el punto, el valor que le muestre en la posición correspondiente de “X”.

Si hacemos esto:

```
archivo_fotogramas[y] = archivo_fotogramas[y] + 0x02;
```

añadimos este valor “0x02” al que ya pudiera tener esta variable. Dicho de otra forma pasamos a encender el LED conectado en esta posición, dejando los demás como estuvieran (A no ser que este LED ya estuviese encendido).



Este es el LED 0x02 de esta fila

Después de dibujar cada punto, mediante las funciones “Impresion_Linea” y “convertir_byte_a_bits” el programa espera una décima de segundo (delay(100);) y borra toda la pantalla para volver a empezar.

La función borrar guarda en todas las variables de la matriz “archivo_fotogramas[ ]” el valor “0x00” por lo que cuando vuelve a la función “moverpunto” el display se encuentra, digamos, limpio para volver a empezar.

Ahora veremos como actúa la suma de estas expresiones en numeración hexadecimal. Si sumamos una expresión con un número hexadecimal en primera posición con otra con un número hexadecimal en segunda posición nos resulta esto, como es lógico.

0x08 + 0x30 = 0x38

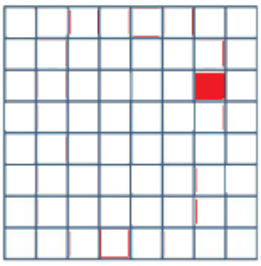
Si sumamos una expresión con un número hexadecimal en segunda posición, con otra expresión con otro número hexadecimal en segunda posición resulta esto, como también es lógico.

0x06 + 0x02 = 0x08

Pero para nosotros ahora esto no es lo importante. Lo interesante es poder ir añadiendo LEDs encendidos a una línea determinada del display a medida que sea necesario. Si en un primer momento “archivo_fotogramas[2]” es igual a “0x00”, si hacemos esto:

archivo_fotogramas[2] = archivo_fotogramas[2] + 0x02;

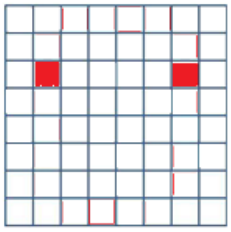
el resultado será el que se puede ver a continuación:



Y si realizo esta operación, sin borrar antes la pantalla:

archivo_fotogramas[2] = archivo_fotogramas[2] + 0x40;

El resultado será este:



Este es el programa completo que mueve un punto por el display, tomando como coordenadas los valores que entran por los potenciómetros conectados en A0 y A1.

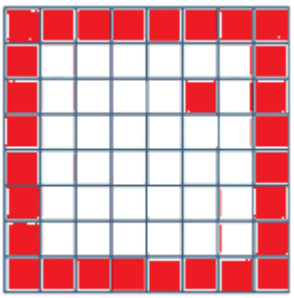
```
int x;
int y;

int indicelinea;
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};      //0

  void convertir_byte_a_bits(char DATA)
 {
    digitalWrite(pinCS,LOW);
       for(indiceLED=8;indiceLED>=1;indiceLED--)
         {
          digitalWrite(pinCLK,LOW);
          digitalWrite(pinDIN,DATA&0x80);  //Extrae cada bit del byte "DATA"
          DATA = DATA<<1;
          digitalWrite(pinCLK,HIGH);
         }
 }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
       digitalWrite(pinCS,LOW);
       convertir_byte_a_bits(indicelinea);    //Envia el valor de la linea del display
       convertir_byte_a_bits(contenidolinea);     //Envia el valor de los LEDs de la
                                                  // linea del display
       digitalWrite(pinCS,HIGH);
}

void puesta_en_marcha()
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01); //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT
                                    // del display :0
}
```

```
void moverpunto()
{
  x = analogRead(A1);
  x = map(x,100,1000,0,7);
  y = analogRead(A0);
  y = map(y,100,1000,0,7);

  Serial.print("Valor de x:");
  Serial.println(x);
  Serial.print("Valor de y:");
  Serial.println(y);


  if (x==0)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x80;
  }

  if (x==1)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x40;
  }

  if (x==2)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x20;
  }

  if (x==3)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x10;
  }

  if (x==4)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x08;
  }

  if (x==5)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x04;
  }

  if (x==6)
  {
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x02;
  }

  if (x==7)
  {
```

```
    archivo_fotogramas[y] = archivo_fotogramas[y] + 0x01;
  }
}


 void borrardisplay()

 {
   archivo_fotogramas[0] = 0x00;
   archivo_fotogramas[1] = 0x00;
   archivo_fotogramas[2] = 0x00;
   archivo_fotogramas[3] = 0x00;
   archivo_fotogramas[4] = 0x00;
   archivo_fotogramas[5] = 0x00;
   archivo_fotogramas[6] = 0x00;
   archivo_fotogramas[7] = 0x00;
 }

void setup()
{
  Serial.begin(9600);
  pinMode(pinCLK,OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(pinDIN,OUTPUT);
  delay(50);
  puesta_en_marcha();
}

void loop()
{
  moverpunto();

        for(indicelinea=1;indicelinea<9;indicelinea++)
       {
       Impresion_Linea(indicelinea, archivo_fotogramas[indicelinea-1]);
       }
      delay(100); //Tiempo de espera entre fotograma y fotograma

  borrardisplay();

}
```

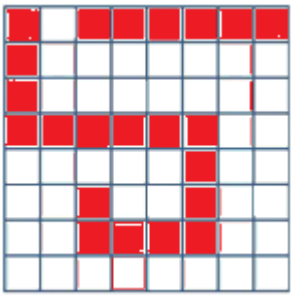
Ejercicio 3

Ahora has de realizar un programa similar al anterior, teniendo en cuenta que el display tenga un marco exterior, de la forma que se puede ver a continuación:

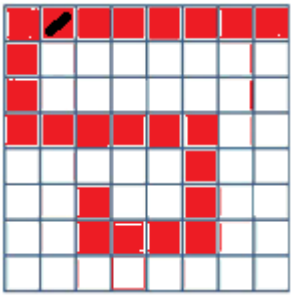


Ejercicio 4

Ahora has de realizar un programa similar al anterior, teniendo en cuenta que el display tenga impreso un laberinto, por donde mover nuestro punto.



Al iniciarse el juego el punto aparecerá en las coordenadas X = 1 e Y = 0, tal como se muestra a continuación:



Ejercicio 5

Se trata de realizar un juego similar al anterior, pero de forma que si el jugador mete el punto por encima de la línea de separación del laberinto, el programa lo detecte y pierda, mostrando un fotograma de “fin del juego”.

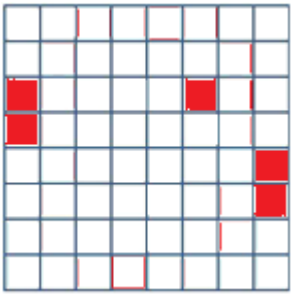
Juego ping pong

Uno de los videojuegos más antiguos y conocidos es el “Pong” que sacó al mercado Atari en noviembre de 1972. Nos proponemos hacer un juego similar sobre el display de 8x8:

Este juego tiene tres elementos, dos paletas, una para cada jugador y una pelota. El juego evoluciona según cómo interactúan los tres elementos.

Las dos paletas se sitúan a la derecha y a la izquierda del display, pudiendo subir y bajar, en la medida que movemos los potenciómetros conectados en A0 y A1. Cada paleta tiene dos puntos en vertical. La pelota es un solo punto que se puede mover hacia adelante y hacia atrás, en horizontal o inclinada hacia arriba o hacia abajo. La pelota rebota en los extremos superior e inferior del display y sobre las paletas (Raquetas). Si la pelota llega al final del lado izquierdo, o derecho, el jugador del lado opuesto se anota un tanto.

El aspecto del campo de juego será este:



Primeramente vamos a dotar de movimiento a una de las paletas, la izquierda. Este será el programa correspondiente:

```
int paletaizquierda;

int indicelinea;
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};//0

  void convertir_byte_a_bits(char DATA)
 {
    digitalWrite(pinCS,LOW);
       for(indiceLED=8;indiceLED>=1;indiceLED--)
         {
          digitalWrite(pinCLK,LOW);
```

```
            digitalWrite(pinDIN,DATA&0x80); //Extrae cada bit del byte "DATA"
            DATA = DATA<<1;
            digitalWrite(pinCLK,HIGH);
           }
  }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
      digitalWrite(pinCS,LOW);
      convertir_byte_a_bits(indicelinea);    //Envia el valor de la linea del display
      convertir_byte_a_bits(contenidolinea);     //Envia el valor de los LEDs de la
                                                 // linea del display
      digitalWrite(pinCS,HIGH);
}

void puesta_en_marcha()
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01); //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT
                                     // del display :0
}


void moverpaletaizquierda()
{
  paletaizquierda = analogRead(A1);
  paletaizquierda = map(paletaizquierda,100,1000,0,6);

  Serial.print("Valor de paletaizquierda:");
  Serial.println(paletaizquierda);

    archivo_fotogramas[paletaizquierda] = archivo_fotogramas[paletaizquierda]
+ 0x80;
    archivo_fotogramas[paletaizquierda+1] =
archivo_fotogramas[paletaizquierda+1] + 0x80;

}

 void borrardisplay()

 {
   archivo_fotogramas[0] = 0x00;
   archivo_fotogramas[1] = 0x00;
   archivo_fotogramas[2] = 0x00;
   archivo_fotogramas[3] = 0x00;
   archivo_fotogramas[4] = 0x00;
   archivo_fotogramas[5] = 0x00;
```

```
   archivo_fotogramas[6] = 0x00;
   archivo_fotogramas[7] = 0x00;
 }

void setup()
{
  Serial.begin(9600);
  pinMode(pinCLK,OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(pinDIN,OUTPUT);
  delay(50);
  puesta_en_marcha();
}

void loop()
{
  moverpaletaizquierda();

        for(indicelinea=1;indicelinea<9;indicelinea++)
       {
       Impresion_Linea(indicelinea, archivo_fotogramas[indicelinea-1]);
       }
      delay(100); //Tiempo de espera entre fotograma y fotograma

  borrardisplay();

}
```

Ejercicio 6

Ahora habrás de completar el programa anterior para que se mueva también la paleta derecha.

Es hora de añadir la pelota

En nuestro juego de “ping pong” añadiremos ahora el elemento correspondiente a la pelota. Comenzaremos por hacer un programa que mueva un punto por la pantalla de forma automática. Este punto será la pelota y se moverá en horizontal hacia adelante.

Creamos una función nueva “pelota_adelante_horizontal()” que va guardando el valor de la variable “archivo_fotogramas[ ]” correspondiente y, además, va incrementando en una unidad el valor de la variable “Xpelota”.

Este es el programa completo que me permite mover la pelota hacia adelante

```
int Xpelota = 0;
int Ypelota = 4;

int indicelinea;
```

```
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};//0

  void convertir_byte_a_bits(char DATA)
 {
    digitalWrite(pinCS,LOW);
       for(indiceLED=8;indiceLED>=1;indiceLED--)
         {
          digitalWrite(pinCLK,LOW);
          digitalWrite(pinDIN,DATA&0x80);  //Extrae cada bit del byte "DATA"
          DATA = DATA<<1;
          digitalWrite(pinCLK,HIGH);
         }
 }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
      digitalWrite(pinCS,LOW);
      convertir_byte_a_bits(indicelinea);    //Envia el valor de la linea del display
      convertir_byte_a_bits(contenidolinea);     //Envia el valor de los LEDs de la
                                                 // linea del display
      digitalWrite(pinCS,HIGH);
}

void puesta_en_marcha()
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01); //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT
                                     // del display :0
}

void pelota_adelante_horizontal()
{

   if (Xpelota==0)
 {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x80;
 }

 if (Xpelota==1)
```

```
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x40;
 }

 if (Xpelota==2)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x20;
 }

 if (Xpelota==3)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x10;
 }

 if (Xpelota==4)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x08;
 }

 if (Xpelota==5)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x04;
 }

 if (Xpelota==6)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x02;
 }

 if (Xpelota==7)
 {
   archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x01;
 }

Xpelota = Xpelota + 1;

}

 void borrardisplay()

 {
   archivo_fotogramas[0] = 0x00;
   archivo_fotogramas[1] = 0x00;
   archivo_fotogramas[2] = 0x00;
   archivo_fotogramas[3] = 0x00;
   archivo_fotogramas[4] = 0x00;
   archivo_fotogramas[5] = 0x00;
   archivo_fotogramas[6] = 0x00;
   archivo_fotogramas[7] = 0x00;
 }
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(pinCLK,OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(pinDIN,OUTPUT);
  delay(50);
  puesta_en_marcha();
}

void loop()
{
  pelota_adelante_horizontal();

        for(indicelinea=1;indicelinea<9;indicelinea++)
       {
       Impresion_Linea(indicelinea, archivo_fotogramas[indicelinea-1]);
       }
      delay(100); //Tiempo de espera entre fotograma y fotograma

  borrardisplay();

}
```

Este programa tiene un problema, cuando llega al extremo de la derecha, sigue incrementando el valor de “Xpelota” por encima de 7 y deja de encender ningún punto, porque Arduino no sabe qué hacer por encima de ese valor.

Para ello debemos incluir un condicional dentro de la función “pelota_adelante_horizontal()” para que haga que ocurra algo cuando el valor de “Xpelota” sea mayor de 7.

Aparte del condicional, crearemos otra función que haga retroceder la pelota hacia atrás en horizontal, de forma que vaya pasando de forma automática de una función a otra rebotando al final del campo, en los dos extremos del mismo, a la derecha y a la izquierda.

Ejercicio 7

Crea un programa que mueva la pelota en horizontal hacia la izquierda, utilizando la función “pelota_atras_horizontal()” y las variables “Xpelota” e “Ypelota”.

Haciendo que rebote la pelota

Ahora que ya hemos creado las dos funciones, una para hacer avanzar la pelota y otra para hacerla retroceder, vamos a hacer que avance y retroceda de forma automática.

Declaramos una nueva variable “indiceFuncion” y dos funciones nuevas “trayectoriapelota()” y “dibujarpelota()”

Del valor que tenga la variable “indiceFuncion” dependerá la trayectoria de la pelota. La selección de la trayectoria se produce en la función “trayectoriapelota()”. Este programa podrá determinar dos trayectorias: hacia adelante o hacia atrás en horizontal. A medida que se vaya completando el programa del juego de “ping pong” se habrán de añadir hacia adelante en diagonal arriba y abajo y hacia atrás en diagonal arriba y abajo. En total se habrán de añadir cuatro trayectorias más.

Gran parte del contenido de las antiguas funciones “pelota_adelante_horizontal()” y “pelota_atras_horizontal()” está ahora en la función “dibujarpelota()”, de forma que se simplifica el programa.

En las funciones “pelota_adelante_horizontal()” y “pelota_atras_horizontal()” se han incluido unos condicionales que hacen variar el valor de la variable “indiceFuncion”, produciendo el cambio de trayectoria.

El programa completo es el siguiente.

```
int Xpelota = - 1;
int Ypelota = 4;
int indiceFuncion = 0;

int indicelinea;
int indiceLED;
//Conexiones entre el display y Arduino
int pinCLK = 10;
int pinCS = 9;
int pinDIN = 8;

char archivo_fotogramas[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};//0

  void convertir_byte_a_bits(char DATA)
 {
    digitalWrite(pinCS,LOW);
       for(indiceLED=8;indiceLED>=1;indiceLED--)
         {
          digitalWrite(pinCLK,LOW);
          digitalWrite(pinDIN,DATA&0x80);  //Extrae cada bit del byte "DATA"
          DATA = DATA<<1;
          digitalWrite(pinCLK,HIGH);
         }
 }

void Impresion_Linea(int indicelinea, char contenidolinea)
{
```

```
        digitalWrite(pinCS,LOW);
        convertir_byte_a_bits(indicelinea);    //Envia el valor de la linea del display
        convertir_byte_a_bits(contenidolinea);     //Envia el valor de los LEDs de la
                                                   // linea del display
        digitalWrite(pinCS,HIGH);
}

void puesta_en_marcha()
{
 Impresion_Linea(0x09, 0x00);       //Tipo de decodificado  : BCD
 Impresion_Linea(0x0a, 0x03);       //Brillo de los LEDs
 Impresion_Linea(0x0b, 0x07);       //Cantidad de LEDs por linea :8 LEDs
 Impresion_Linea(0x0c, 0x01); //Desconexion del display  : 0   Modo normal  : 1
 Impresion_Linea(0x0f, 0x00);       //Modo de prueba del display :1   Modo EOT
                                     // del display :0
}


 void trayectoriapelota()
 {

   if (indiceFuncion==0)
  {
    pelota_adelante_horizontal();
  }

  if (indiceFuncion==1)
  {
    pelota_atras_horizontal();
  }
 }

 void dibujarpelota()
 {
     if (Xpelota==0)
  {
     archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x80;
  }

  if (Xpelota==1)
  {
     archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x40;
  }

  if (Xpelota==2)
  {
     archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x20;
  }

  if (Xpelota==3)
```

```
  {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x10;
  }

  if (Xpelota==4)
  {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x08;
  }

  if (Xpelota==5)
  {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x04;
  }

  if (Xpelota==6)
  {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x02;
  }

  if (Xpelota==7)
  {
    archivo_fotogramas[Ypelota] = archivo_fotogramas[Ypelota] + 0x01;
  }
 }

 void pelota_adelante_horizontal()
{

Xpelota = Xpelota + 1;

if (Xpelota==8)
{
  indiceFuncion = 1;
}

}

void pelota_atras_horizontal()
{

Xpelota = Xpelota - 1;

if (Xpelota==0)
{
  indiceFuncion = 0;
}

}

 void borrardisplay()
```

```
 {
   archivo_fotogramas[0] = 0x00;
   archivo_fotogramas[1] = 0x00;
   archivo_fotogramas[2] = 0x00;
   archivo_fotogramas[3] = 0x00;
   archivo_fotogramas[4] = 0x00;
   archivo_fotogramas[5] = 0x00;
   archivo_fotogramas[6] = 0x00;
   archivo_fotogramas[7] = 0x00;
 }

void setup()
{
  Serial.begin(9600);
  pinMode(pinCLK,OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(pinDIN,OUTPUT);
  delay(50);
  puesta_en_marcha();
}

void loop()
{

  trayectoriapelota();

  dibujarpelota();

        for(indicelinea=1;indicelinea<9;indicelinea++)
       {
       Impresion_Linea(indicelinea, archivo_fotogramas[indicelinea-1]);
       }
      delay(100); //Tiempo de espera entre fotograma y fotograma

  borrardisplay();

}
```

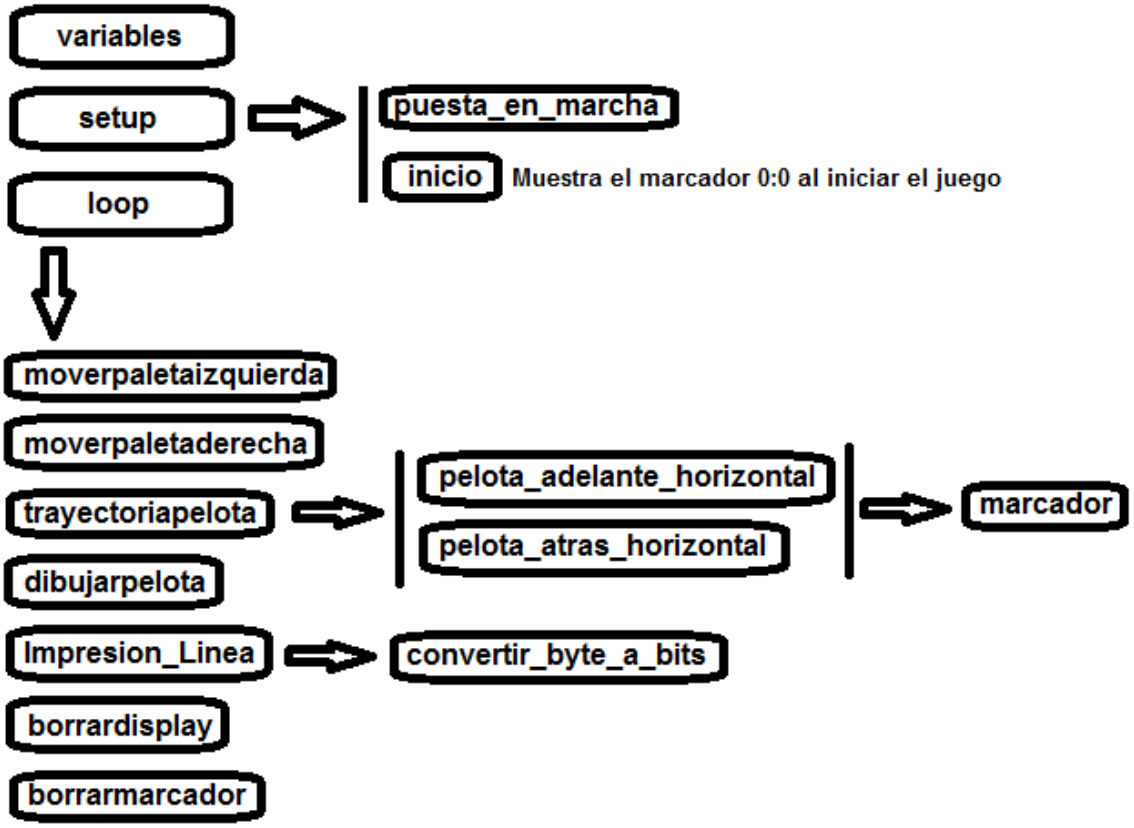
Haciendo que rebote la pelota en horizontal con las dos paletas

Sobre el programa anterior vamos a recuperar la parte del código que nos permite mover las dos paletas con los valores de los dos potenciómetros conectados en A0 y A1.

La pelota ahora ha de actuar de forma diferente. Si llega a la paleta rebota, pero si no toca la paleta y llega al fondo del campo, el jugador del otro extremo se anota un tanto, que también podremos mostrar en la pantalla.

Para guardar los fotogramas de los diferentes valores del marcador y otras imágenes que sean necesarias para el juego, volveremos a usar una matriz de doble entrada en “archivo_fotogramas[ ] [ ]”. Para manejar estos fotogramas crearemos la variable “indiceFotograma”.

El diagrama de bloques se muestra a continuación. Se ha creado la función “inicio()” que se ejecuta al principio desde el “setup” y que muestra al inicio del juego el marcador con la puntuación 0:0, espera un pequeño tiempo y da inicio al juego con la aparición de las paletas y la pelota.



La función “marcador()” muestra el fotograma correspondiente al marcador con la puntuación que corresponda. A esta función se llega a partir de las funciones correspondientes a las diferentes trayectorias de la pelota (“pelota_adelante_horizontal” y “pelota_atras_horizontal”). Cuando el juego esté terminado también han de existir las funciones “adelante_arriba”, “adelante_abajo”, atras_arriba” y “atras_abajo”.

Si la pelota va hacia la izquierda y toca la paleta, rebotará hacia adelante y esto lo podemos conseguir con el siguiente condicional.

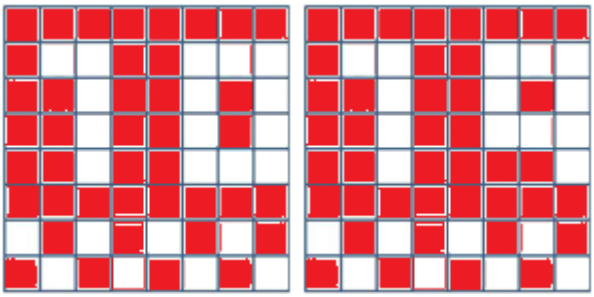
```
if ((Xpelota==1)&&((Ypelota==paletaizquierda)||(Ypelota==paletaizquierda+1)))
{
  indiceFuncion = 0;
}
```

Si la pelota va hacia la derecha y llega al final del campo, aumentamos el valor del marcador y nos dirigimos a la función “marcador()”.

Esto se puede conseguir con el siguiente condicional.

```
if (Xpelota==8)
{
  marcadorizquierdo = marcadorizquierdo + 1;
 marcador();
}
```

Los fotogramas del marcador pueden ser similares a estos.



Para actualizar el fotograma del marcador basta con que la función “marcador()” actualice la parte correspondiente al jugador que se haya anotado el tanto. Esto se puede conseguir de la siguiente forma:

```
  if (marcadorderecho==2)
{
   archivo_fotogramas[1][1] = archivo_fotogramas[1][1] + 0x00;
   archivo_fotogramas[1][2] = archivo_fotogramas[1][2] + 0x06;
   archivo_fotogramas[1][3] = archivo_fotogramas[1][3] + 0x05;
   archivo_fotogramas[1][4] = archivo_fotogramas[1][4] + 0x00;
}
```

Ejercicio 8

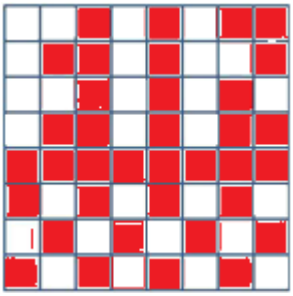
Completa el programa comentado anteriormente. Recuerda que en él se mueven las dos paletas, la pelota se puede mover adelante y atrás, cuando la pelota toca la paleta rebota, al iniciarse el juego se muestra el marcador 0:0 y siempre que la pelota llega al final del campo, el jugador que la ha lanzado se anota un tanto que se muestra de nuevo en el marcador.

Ejercicio 9

Ahora hemos de lograr que cuando el valor del marcador llegue a 10, para cualquier jugador, como que no nos cabe un valor tan alto en el display, finalice el juego y se reinicie.

Para ello crearemos una nueva función llamada “nuevojuego()” al que se llega desde la función “marcador()”. En esta función nuevo juego se muestra un fotograma que indica que se ha finalizado el juego y se ponen a cero todas las variables que lo necesiten para iniciar un nuevo juego.

Esta puede ser la imagen del final del juego.



Conexión de dos displays

Arduino puede manejar más de un display, para ello se han de conectar de la siguiente manera.