Laboratorio de microcontroladores.
Alejando Ossio Díaz, A01209122.
Gerardo Daniel Naranjo Gallegos, A01209499.
Roberto Figueroa Saavedra, A01209689.
27 de octubre de 2016.
Práctica #5.

Matriz de ledes.

Objetivo:

En una matriz de ledes desplegar mensajes y animaciones, con ayuda del micro, un MAX7219 y un MSGEQ7.

Descripción:

La práctica consiste en cuatro partes. Todas corresponden al uso de la matriz de ledes con el microcontrolador.

Mensaje:

La primera parte es desplegar un mensaje, con un mínimo de 10 letras, en donde irá apareciendo letra por letra en el display.

Mensaje con corrimiento:

Esta parte de la práctica consiste en desplegar otro mensaje, de mínimo 10 letras, pero ahora con corrimiento de letras hacia la izquierda.

Animación:

La parte de animación corresponde a mostrar en la matriz de ledes una animación creada por nosotros y utilizando al menos 3 diferentes niveles de intensidad de luz en la matriz.

La variación de la intensidad de luz se puede dar de manera analógico o digital. Para hacerlo de manera analógica, podemos enviar distintos valores de voltaje entre 3.3V y 5V; entre más alto el voltaje, mayor la intensidad de luz. Para hacerlo de manera digital, dependerá del valor que mandemos en el registro del MAX7219, como podemos ver en la siguiente tabla:

Table 7. Intensity Register Format (Address (Hex) = 0xXA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	0xX0
3/32	2/16	X	X	X	X	0	0	0	1	0xX1
5/32	3/16	X	X	X	X	0	0	1	0	0xX2
7/32	4/16	X	X	X	X	0	0	1	1	0xX3
9/32	5/16	X	X	X	X	0	1	0	0	0xX4
11/32	6/16	X	X	X	X	0	1	0	1	0xX5
13/32	7/16	X	X	X	X	0	1	1	0	0xX6
15/32	8/16	X	X	X	X	0	1	1	1	0xX7
17/32	9/16	X	X	X	X	1	0	0	0	0xX8
19/32	10/16	X	X	X	X	1	0	0	1	0xX9
21/32	11/16	X	X	X	X	1	0	1	0	0xXA
23/32	12/16	X	X	X	X	1	0	1	1	0xXB
25/32	13/16	X	X	X	X	1	1	0	0	0xXC
27/32	14/16	X	X	X	X	1	1	0	1	0xXD
29/32	15/16	X	X	X	X	1	1	1	0	0xXE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	0xXF

(MAXIM, 2016)

Ecualizador:

Se debe implementar un ecualizador en la matriz de ledes, mostrando la intensidad de cada rango de frecuencias que da el circuito MSGEQ7, las cuales son 63Hz, 160Hz, 400Hz, 1kHz, 2.5kHz, 6.25kHz y 16kHz y utilizando una columna de ledes para cada una de las frecuencias.

Circuito:

El circuito consiste en 3 *push-button*, que tienen la función de seleccionar entre las funciones (mensaje, mensaje con corrimiento y animación) al presionar cada uno de ellos.

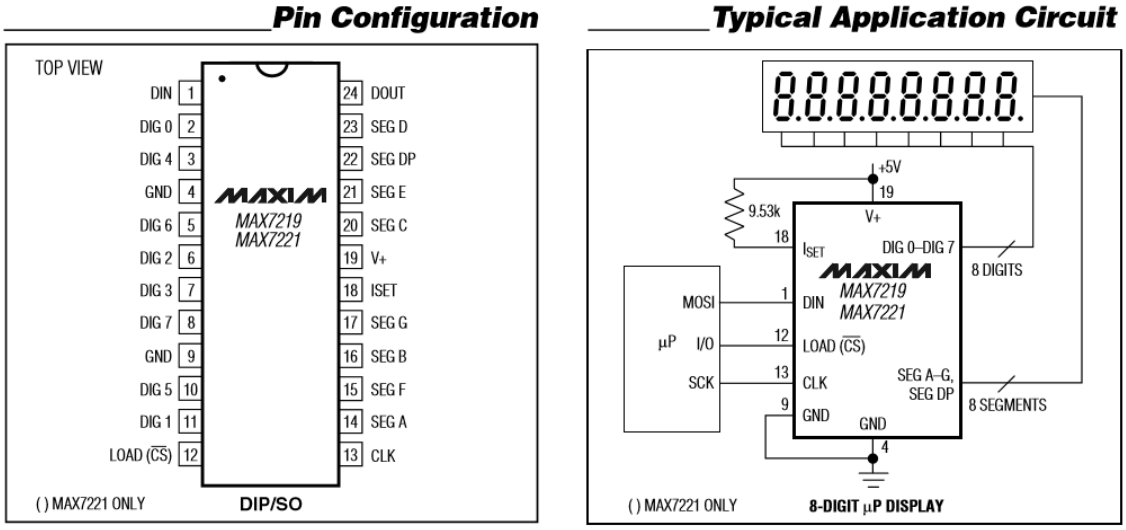
También, un *switch* que nos permitirá seleccionar si el modo ecualizador está encendido o apagado. Es decir, mientras esté encendido, la matriz de ledes mostrará el ecualizador y, al estar apagado, se podrá escoger qué función mostrar entre el mensaje, mensaje con corrimiento o animación, con la ayuda de los *push-button*.

Para la utilización de la matriz de ledes necesitamos también un MAX7219, que nos permitirá controlar la matriz. El MAX7219 cuenta con 16 salidas, divididas en 8 segmentos y 8 dígitos, que ayudan a controlar la matriz de ledes, ya que de estos pines se manda la información a la

matriz. Cuenta también con un puerto llamado DOUT que nos permite mandar datos de manera serial.

En cuanto a las entradas, cuenta con un DIN para recibir la información de manera serial; un CLK para la entrada serial del *clock*, ISET para regular la corriente pico al conectarse a voltaje mediante una resistencia; la entrada LOAD/CS que nos permite mandar datos de manera serial a un *latch* hasta el siguiente pulso (arriba) del reloj; y dos tierras y una entrada de voltaje.

Podemos ver los pines del MAX7219 en las siguientes ilustraciones:



(MAXIM, 2016)

Para la función del ecualizador, necesitaremos un MSGEQ7, que nos ayudará a determinar las frecuencias (63Hz, 160Hz, 400Hz, 1kHz, 2.5kHz, 6.25kHz and 16kHz) de una canción.

Código:

En cuanto al código, se deberán implementar distintas funciones:

La función MandaMax7219(unsigned int dato), que recibe los datos (16 bits) a transmitir, de manera serial, al MAX7219. Para esto, se deberá leer primero el protocolo de comunicación en la hoja de especificaciones.

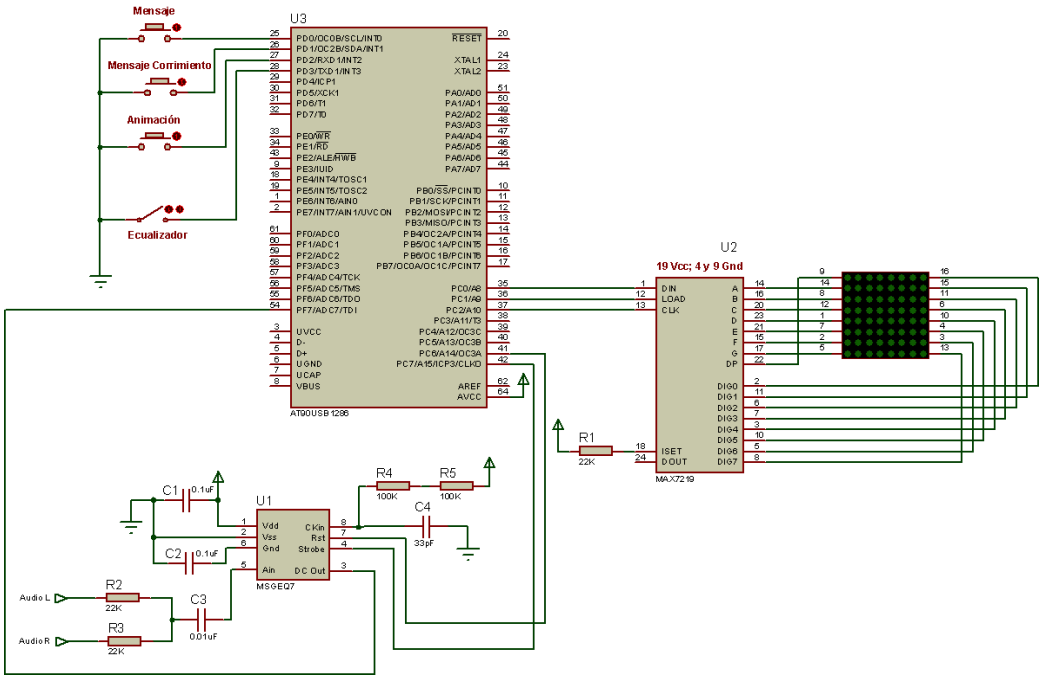
Incluir también la función es ConfiguraMax(). El procedimiento configura el MAX7219. Los registros a configurar para nuestra aplicación son: el 0x09, 0x0A, 0x0B, 0x0C y 0x0F.

Otra función a implementar es ConfiguraIntensidad(unsigned char c), que cambia la luminosidad de los ledes.

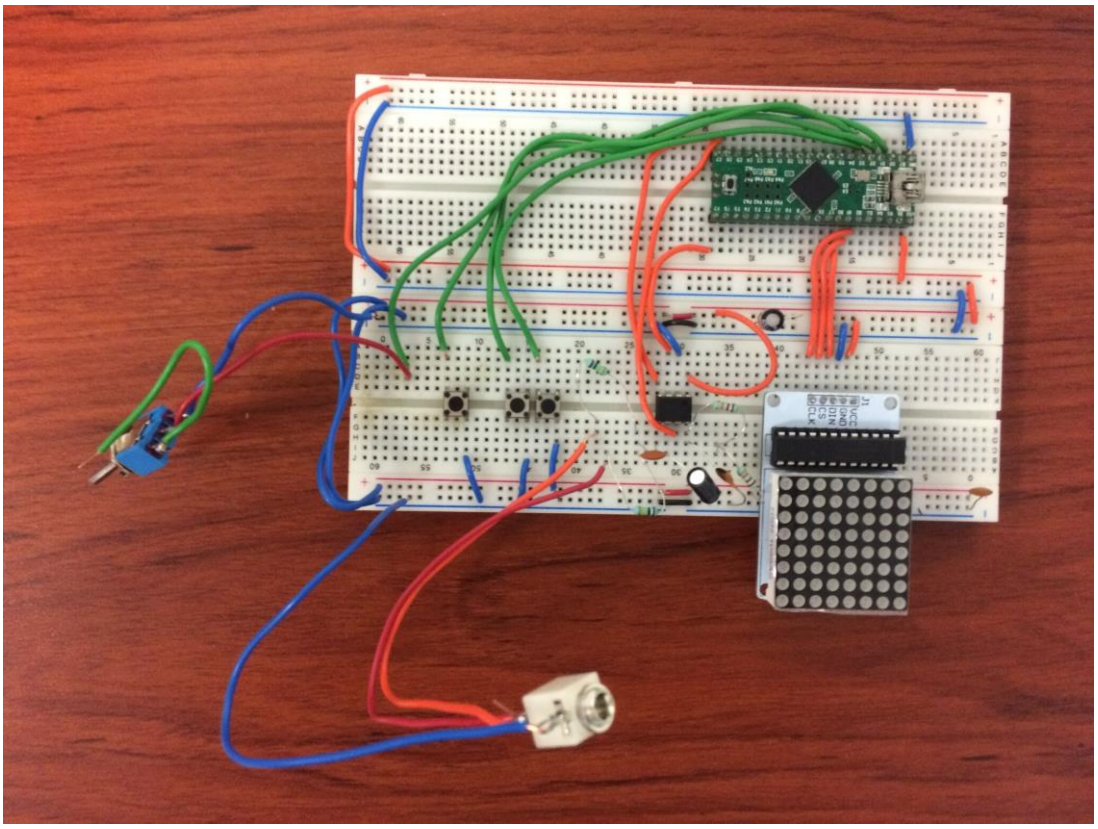
También, tendremos la función DespliegaLetra(unsigned char c). Esta función, recibe la letra a desplegar en la matriz, para acceder a la tabla llamada "TablaLetras.txt" y generarla. Nota: entre el primer elemento de la tabla y su correspondiente en ASCII existe un desfasamiento en 32.

Por último, se deberá incluir la función DespliegaMensaje(unsigned char Mensaje[ ], unsigned int tiempo) y la función DespliegaMensajeCorrimiento(unsigned char Mensaje[ ], unsigned int tiempo). Ambas son funciones que reciben un string con el mensaje que se desea mostrar en la matriz de ledes y un entero. Para la primera función, el entero representa la cantidad de milisegundos que tardará en mandar cada letra. Por otro lado, para la segunda función, el entero representa la cantidad de milisegundos que tardará en mandar cada letra, pero se deben ir recorriendo las letras (es decir, aplicar el corrimiento del mensaje).

Diagrama eléctrico:



Fotografía del circuito:



Código fuente:

El código fuente utilizado en el microcontrolador es el siguiente:

```
#include <90USB1286.h>
#include <90USB1286.h>
 #include <stdio.h>
 #include <stdlib.h>
 #include <delay.h>
 #define DIN_1  PORTB.0
 #define LOAD PORTB.1
 #define CLK  PORTB.2
 #define RST  PORTC.6
 #define STR  PORTC.7


#define ADC_VREF_TYPE 0x40

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
```

```
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

 int v1,v2,v3,v4,v5,v6,v7,v8,j,a,i,b,c,d,e,g;
 int d1,d2,d3,d4,d5,d6,d7,d8;
 unsigned int e1,e2,e3,e4,e5,e6,e7;

flash char Letras[133][5] = {{0,0,0,0,0},    //espacio
    {0,0,242,0,0        },    // !
    {0,224,0,224,0      },    // "
    {40,254,40,254,40   },    // #
    {36,84,254,84,72    },    // $
    {196,200,16,38,70   },    // %
    {108,146,170,132,10 },    // &
    {0,160,192,0,0      },    // ï
    {0,56,68,130,0      },    // (
    {0,130,68,56,0      },    // )
    {40,16,124,16,40    },    // *
    {16,16,124,16,16    },    // +
    {0,5,6,0,0          },    // ,
    {16,16,16,16,16     },    // -
    {0,6,6,0,0          },    // .
    {4,8,16,32,64       },    // /
    {124,138,146,162,124},    // 0
    {0,66,254,2,0       },    // 1
    {66,134,138,146,98  },    // 2
    {132,130,162,226,156},    // 3
    {24,40,72,254,8     },    // 4
    {228,162,162,162,156},    // 5
    {60,82,146,146,12   },    // 6
    {128,142,144,160,192},    // 7
    {108,146,146,146,108},    // 8
    {96,146,146,148,120 },    // 9
    {0,54,54,0,0        },    // :
    {0,53,54,0,0        },    // ;
    {16,40,68,130,0     },    // <
    {40,40,40,40,40     },    // =
    {0,130,68,40,16     },    // >
    {64,128,138,144,96  },    // ?
    {76,146,158,130,124 },    // @
    {126,144,144,144,126},    // A   //33
    {254,146,146,146,108},    // B
    {124,130,130,130,68 },    // C
    {254,130,130,130,124},    // D
    {254,146,146,146,146},    // E
    {254,144,144,144,144},    // F
    {124,130,146,146,92 },    // G
    {254,16,16,16,254   },    // H
```

```
{0,130,254,130,0    },    // I
{132,130,130,130,252},    // J
{254,16,40,68,130   },    // K
{254,2,2,2,2        },    // L
{254,64,32,64,254   },    // M
{254,64,32,16,254   },    // N
{124,130,130,130,124},    // O
{254,144,144,144,96 },    // P
{124,130,134,130,125},    // Q
{254,144,152,148,98 },    // R
{100,146,146,146,76 },    // S
{128,128,254,128,128},    // T
{252,2,2,2,252      },    // U
{248,4,2,4,248      },    // V
{254,4,8,4,254      },    // W
{198,40,16,40,198   },    // X
{192,32,30,32,192   },    // Y
{134,138,146,162,194},    // Z     //58
{0,254,130,130,0    },    // [
{64,32,16,8,4       },    // \
{0,130,130,254,0    },    // ]
{32,64,128,64,32    },    // ^
{2,2,2,2,2          },    // _
{0,128,64,32,0      },    // `
{4,42,42,42,30      },    // a          //65
{254,18,34,34,28    },    // b
{28,34,34,34,34     },    // c
{28,34,34,18,254    },    // d
{28,42,42,42,24     },    // e
{16,126,144,128,64  },    // f
{24,37,37,37,62     },    // g
{254,16,32,32,30    },    // h
{0,34,190,2,0       },    // i
{4,2,34,188,0       },    // j
{254,8,20,34,0      },    // k
{0,130,254,2,0      },    // l
{62,32,24,32,30     },    // m
{62,16,32,32,30     },    // n
{28,34,34,34,28     },    //
{63,36,36,36,24     },    // p
{24,36,36,36,63     },    // q
{62,16,32,32,16     },    // r
{18,42,42,42,4      },    // s
{32,254,34,2,4      },    // t
{60,2,2,4,62        },    // u
{56,4,2,4,56        },    // v
{60,2,12,2,60       },    // w
{34,20,8,20,34      },    // x
{48,10,10,10,60     },    // y
{34,38,42,50,34     },    // z          //90
{0,16,108,130,0     },    // {
```

```
   {0,0,254,0,0        },    // |
   {0,130,108,16,0     },    // }
   {16,32,16,8,16      },    // ~
   {0,0,0,0,0          },    //;
   {4,170,42,170,30    },    //„
   {28,162,34,162,28   },    //
   {28,66,2,66,28      },    // ?
   {62,208,80,208,62   },    // Ž
   {60,194,66,194,60   },     // ™
   {60,130,2,130,60    },     // š
   {127,144,146,146,108}};     // á
char Msg[]  = "LabdeMicros";
flash char Pac1[]={60, 126, 255, 255, 191, 231, 66, 0};
flash char Pac2[]={60, 126, 255, 255, 191, 247, 98, 0};
flash char Pac3[]={60, 126, 255, 255, 191, 255, 126, 0};
unsigned int Audio[7];

void MandaMax7219_1(unsigned int dato){
   unsigned char i;
   CLK=0;
   LOAD=0;
   //ciclo para mandar bit por bit (serial)
   for(i=0;i<16;i++)
   {
      if((dato&0x8000)==0)
         DIN_1=0;
      else
         DIN_1=1;
      CLK=1;
      CLK=0;
      dato=dato<<1; //esto es lo mismo a dividirlo entre 2, para recorrer datos
   }
   LOAD=1;
   LOAD=0;
}

//Procedimeinto que se tiene que llamar de inicio para poder controlar la
matriz de leds
void ConfiguraMax_1(void){
   DDRB |=0x07;
   MandaMax7219_1(0x0900);
   MandaMax7219_1(0x0A0B);
   MandaMax7219_1(0x0B07);
   MandaMax7219_1(0x0C01);
   MandaMax7219_1(0x0F00);
}

//Configura la intensidad
void ConfiguraIntensidad(unsigned char c){
switch (c){
case 0:
```

```
MandaMax7219_1(0x0A00);
break;
case 1:
MandaMax7219_1(0x0A01);
break;
case 2:
MandaMax7219_1(0x0A02);
break;
case 3:
MandaMax7219_1(0x0A03);
break;
case 4:
MandaMax7219_1(0x0A04);
break;
case 5:
MandaMax7219_1(0x0A05);
break;
case 6:
MandaMax7219_1(0x0A06);
break;
case 7:
MandaMax7219_1(0x0A07);
break;
case 8:
MandaMax7219_1(0x0A08);
break;
case 9:
MandaMax7219_1(0x0A09);
break;
case 10:
MandaMax7219_1(0x0A0A);
break;
case 11:
MandaMax7219_1(0x0A0B);
break;
case 12:
MandaMax7219_1(0x0A0C);
break;
case 13:
MandaMax7219_1(0x0A0D);
break;
case 14:
MandaMax7219_1(0x0A0E);
break;
case 15:
MandaMax7219_1(0x0A0F);
break;
}
}

void DespliegaLetra(unsigned char c){
```

```
    int valor;
    valor=c-32;
        v1=Letras[valor][0];
        v2=Letras[valor][1];
        v3=Letras[valor][2];
        v4=Letras[valor][3];
        v5=Letras[valor][4];

        d1=(7*256)+v1;
        d2=(6*256)+v2;
        d3=(5*256)+v3;
        d4=(4*256)+v4;
        d5=(3*256)+v5;
       MandaMax7219_1(d1);
       MandaMax7219_1(d2);
       MandaMax7219_1(d3);
       MandaMax7219_1(d4);
       MandaMax7219_1(d5);
    }


void DespliegaMensaje(unsigned char Mensaje[ ], unsigned int tiempo){
unsigned char x=sizeof(Msg)-1;
   for(i=0;i<x;i++){
      DespliegaLetra(Mensaje[i]);
      delay_ms(tiempo);
   }
}

void DespliegaMensajeCor(unsigned char Mensaje[ ], unsigned int tiempo){
    int valor2;
         unsigned char x=sizeof(Msg)-1;

    i=0;
    for(i=0;i<x;i++){
       valor2=Mensaje[i]-32;
        a=Letras[valor2][0];
        b=Letras[valor2][1];
        c=Letras[valor2][2];
        d=Letras[valor2][3];
        e=Letras[valor2][4];

        for( g=1; g<(9); g++){
          d1=(g*256)+a;
                  MandaMax7219_1(d1);

          d2=((g-1)*256)+b;
                  MandaMax7219_1(d2);

          d3=((g-2)*256)+c;
                  MandaMax7219_1(d3);
```

```
        d4=((g-3)*256)+d;
                MandaMax7219_1(d4);

        d5=((g-4)*256)+e;
                MandaMax7219_1(d5);

        delay_ms(tiempo);

        MandaMax7219_1(0x0100);
        MandaMax7219_1(0x0200);
        MandaMax7219_1(0x0300);
        MandaMax7219_1(0x0400);
    }

    for(g=1;g<2;g++){
    MandaMax7219_1(8*256+b);
    MandaMax7219_1(7*256+c);
    MandaMax7219_1(6*256+d);
    MandaMax7219_1(5*256+e);
    delay_ms(tiempo);
     MandaMax7219_1(0x0500);

    }

    for(g=1;g<2;g++){
    MandaMax7219_1(8*256+c);
    MandaMax7219_1(7*256+d);
    MandaMax7219_1(6*256+e);
          delay_ms(tiempo);

     MandaMax7219_1(0x0600);

    }
    for(g=1;g<2;g++){
    MandaMax7219_1(8*256+d);
    MandaMax7219_1(7*256+e);
          delay_ms(tiempo);

     MandaMax7219_1(0x0700);

    }
    for(g=1;g<2;g++){
    MandaMax7219_1(8*256+e);
          delay_ms(tiempo);

     MandaMax7219_1(0x0800);

    }

    }
```

```
}

void Animacion1(){
   ConfiguraIntensidad(15);

      v1=Pac1[0];
      v2=Pac1[1];
      v3=Pac1[2];
      v4=Pac1[3];
      v5=Pac1[4];
      v6=Pac1[5];
      v7=Pac1[6];
      v8=Pac1[7];
      d1=(8*256)+v1;
      d2=(7*256)+v2;
      d3=(6*256)+v3;
      d4=(5*256)+v4;
      d5=(4*256)+v5;
      d6=(3*256)+v6;
      d7=(2*256)+v7;
      d8=(1*256)+v8;
      MandaMax7219_1(d1);
      MandaMax7219_1(d2);
      MandaMax7219_1(d3);
      MandaMax7219_1(d4);
      MandaMax7219_1(d5);
      MandaMax7219_1(d6);
      MandaMax7219_1(d7);
      MandaMax7219_1(d8);

}

void Animacion2(){
   ConfiguraIntensidad(6);

      v1=Pac2[0];
      v2=Pac2[1];
      v3=Pac2[2];
      v4=Pac2[3];
      v5=Pac2[4];
      v6=Pac2[5];
      v7=Pac2[6];
      v8=Pac2[7];
      d1=(8*256)+v1;
      d2=(7*256)+v2;
      d3=(6*256)+v3;
      d4=(5*256)+v4;
      d5=(4*256)+v5;
      d6=(3*256)+v6;
      d7=(2*256)+v7;
```

```
        d8=(1*256)+v8;
        MandaMax7219_1(d1);
        MandaMax7219_1(d2);
        MandaMax7219_1(d3);
        MandaMax7219_1(d4);
        MandaMax7219_1(d5);
        MandaMax7219_1(d6);
        MandaMax7219_1(d7);
        MandaMax7219_1(d8);

}

void Animacion3(){
   ConfiguraIntensidad(1);

        v1=Pac3[0];
        v2=Pac3[1];
        v3=Pac3[2];
        v4=Pac3[3];
        v5=Pac3[4];
        v6=Pac3[5];
        v7=Pac3[6];
        v8=Pac3[7];
        d1=(8*256)+v1;
        d2=(7*256)+v2;
        d3=(6*256)+v3;
        d4=(5*256)+v4;
        d5=(4*256)+v5;
        d6=(3*256)+v6;
        d7=(2*256)+v7;
        d8=(1*256)+v8;
        MandaMax7219_1(d1);
        MandaMax7219_1(d2);
        MandaMax7219_1(d3);
        MandaMax7219_1(d4);
        MandaMax7219_1(d5);
        MandaMax7219_1(d6);
        MandaMax7219_1(d7);
        MandaMax7219_1(d8);
}

void Ani(){
   Animacion1();
   delay_ms(200);
   Animacion2();
   delay_ms(200);
   Animacion3();
   delay_ms(200);
   Animacion2();
   delay_ms(200);
   Animacion1();
```

```
}

void Ecualizador(){
RST=1;
STR=1;
delay_us(50);
STR=0;
delay_us(50);
RST=0;
STR=1;
delay_us(50);
i=0;
for(i=0;i<7;i++){                //leer 7 bandas
   STR=0;
   delay_us(40);
   Audio[i]=read_adc(7);
   delay_us(40);
   STR=1;
   delay_us(80);
}
      e1=Audio[0]/128;
      e2=Audio[1]/128;
      e3=Audio[2]/128;
      e4=Audio[3]/128;
      e5=Audio[4]/128;
      e6=Audio[5]/128;
      e7=Audio[6]/128;

     // 63
     switch(e1){

      case 1:
      MandaMax7219_1(256*8+e1);
      break;
      case 2:
      MandaMax7219_1(256*8+e1+1);
      break;
      case 3:
      MandaMax7219_1(256*8+e1+4);
      break;
      case 4:
      MandaMax7219_1(256*8+e1+11);
      break;
      case 5:
      MandaMax7219_1(256*8+e1+26);
      break;
      case 6:
      MandaMax7219_1(256*8+e1+57);
      break;
      case 7:
      MandaMax7219_1(256*8+e1+120);
```

```
      break;
      case 8:
      MandaMax7219_1(256*8+e1+247);
      break;

    }
     // 160
    switch(e2){
     case 1:
     MandaMax7219_1(256*7+e2);
     break;
     case 2:
     MandaMax7219_1(256*7+e2+1);
     break;
     case 3:
     MandaMax7219_1(256*7+e2+4);
     break;
     case 4:
     MandaMax7219_1(256*7+e2+11);
     break;
     case 5:
     MandaMax7219_1(256*7+e2+26);
     break;
     case 6:
     MandaMax7219_1(256*7+e2+57);
     break;
     case 7:
     MandaMax7219_1(256*7+e2+120);
     break;
     case 8:
     MandaMax7219_1(256*7+e2+247);
     break;

    }
    // 400
    switch(e3){
     case 1:
     MandaMax7219_1(256*6+e3);
     break;
     case 2:
     MandaMax7219_1(256*6+e3+1);
     break;
     case 3:
     MandaMax7219_1(256*6+e3+4);
     break;
     case 4:
     MandaMax7219_1(256*6+e3+11);
     break;
     case 5:
     MandaMax7219_1(256*6+e3+26);
     break;
```

```
      case 6:
      MandaMax7219_1(256*6+e3+57);
      break;
      case 7:
      MandaMax7219_1(256*6+e3+120);
      break;
      case 8:
      MandaMax7219_1(256*6+e3+247);
      break;

    }
    // 1k
    switch(e4){
      case 1:
      MandaMax7219_1(256*5+e4);
      break;
      case 2:
      MandaMax7219_1(256*5+e4+1);
      break;
      case 3:
      MandaMax7219_1(256*5+e4+4);
      break;
      case 4:
      MandaMax7219_1(256*5+e4+11);
      break;
      case 5:
      MandaMax7219_1(256*5+e4+26);
      break;
      case 6:
      MandaMax7219_1(256*5+e4+57);
      break;
      case 7:
      MandaMax7219_1(256*5+e4+120);
      break;
      case 8:
      MandaMax7219_1(256*5+e4+247);
      break;

    }
    // 2.5k
    switch(e5){
      case 1:
      MandaMax7219_1(256*4+e5);
      break;
      case 2:
      MandaMax7219_1(256*4+e5+1);
      break;
      case 3:
      MandaMax7219_1(256*4+e5+4);
      break;
      case 4:
```

```
      MandaMax7219_1(256*4+e5+11);
      break;
      case 5:
      MandaMax7219_1(256*4+e5+26);
      break;
      case 6:
      MandaMax7219_1(256*4+e5+57);
      break;
      case 7:
      MandaMax7219_1(256*4+e5+120);
      break;
      case 8:
      MandaMax7219_1(256*4+e5+247);
      break;

    }
    // 6.25k
    switch(e6){
     case 1:
     MandaMax7219_1(256*3+e6);
     break;
     case 2:
     MandaMax7219_1(256*3+e6+1);
     break;
     case 3:
     MandaMax7219_1(256*3+e6+4);
     break;
     case 4:
     MandaMax7219_1(256*3+e6+11);
     break;
     case 5:
     MandaMax7219_1(256*3+e6+26);
     break;
     case 6:
     MandaMax7219_1(256*3+e6+57);
     break;
     case 7:
     MandaMax7219_1(256*3+e6+120);
     break;
     case 8:
     MandaMax7219_1(256*3+e6+247);
     break;

    }
    // 16|k
    switch(e7){
     case 1:
     MandaMax7219_1(256*2+e7);
     break;
     case 2:
     MandaMax7219_1(256*2+e7+1);
```

```
        break;
        case 3:
        MandaMax7219_1(256*2+e7+4);
        break;
        case 4:
        MandaMax7219_1(256*2+e7+11);
        break;
        case 5:
        MandaMax7219_1(256*2+e7+26);
        break;
        case 6:
        MandaMax7219_1(256*2+e7+57);
        break;
        case 7:
        MandaMax7219_1(256*2+e7+120);
        break;
        case 8:
        MandaMax7219_1(256*2+e7+247);
        break;

      }
        switch(e7){
        case 1:
        MandaMax7219_1(256*1+e7);
        break;
        case 2:
        MandaMax7219_1(256*1+e7+1);
        break;
        case 3:
        MandaMax7219_1(256*1+e7+4);
        break;
        case 4:
        MandaMax7219_1(256*1+e7+11);
        break;
        case 5:
        MandaMax7219_1(256*1+e7+26);
        break;
        case 6:
        MandaMax7219_1(256*1+e7+57);
        break;
        case 7:
        MandaMax7219_1(256*1+e7+120);
        break;
        case 8:
        MandaMax7219_1(256*1+e7+247);
        break;

      }
}

void main(){}
```

Conclusiones individuales:

Alejandro Ossio:

En esta práctica se utilizó una matriz de leds de 8x8, pero como yo la habíamos usado anteriormente no se complicó esta práctica, también sirvió para recordar el conversor analógico digital que tiene el micro controlador.

Gerardo Naranjo:

La práctica consiste en el uso de una matriz de ledes, cosa que ya habíamos hecho para nuestro proyecto final de la materia de microcontroladores, por lo tanto, ya estábamos familiarizados con el tema y no fue mayor complicación realizar la práctica. Nos fue de utilidad para recordar el manejo de una matriz de ledes con el micro y la conversión analógico digital. Por último, resultó interesante realizar el ecualizador.

Roberto Figueroa:

Con esta práctica creamos una librería para interactuar con una matriz de leds, la cual ya habíamos manejado previamente en el semestre anterior, por lo cual no hubo gran problema al elaborarla, lo nuevo que desarrollos fue del ecualizador y el chip con el que leíamos sus valores de entrada para así poder mandarlo al display.

Referencias:

MAXIM. (2016). *Serially Interfaced, 8-Digit LED Display Drivers*. Obtenido de MAX7219: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf