Alejandro Ossio Díaz, A01209122.
Gerardo Daniel Naranjo Gallegos, A01209499.
Roberto Saavedra Figueroa, A01209689.
Profesor: Agustín Domínguez Oviedo.
Curso de microcontroladores.
ITESM, campus Querétaro.
18 de mayo de 2016.

Proyecto final: reloj, con segundero en binario y alarma.

Objetivo de la práctica:

Nuestro objeto fue hacer un reloj, utilizando matrices de ledes para mostrar las horas y minutos; los segundos se mostrarán en binario mediante un *display* de ledes. También contará con la opción de ajustar la hora y de fijar una alarma.

Descripción de la práctica.

Esta práctica pretende programar un microcontrolador ATMega328P y crear un circuito, en donde se implemente un reloj con alarma. Primero, probando el código de manera virtual en Proteus, para tener certeza de su funcionamiento; después, probándolo físicamente en el circuito. El código a programar en el ATMega328P se encuentra más adelante en este reporte. Se utilizaron interrupciones, para generar un segundo exacto.

Los componentes utilizados en el circuito son: un microcontrolador ATMega328P, cuatro matrices de ledes, cuatro MAX7219 (para el control de las matrices), un *display* de ledes, una bocina, cuatro botones, cinco *switches* y varios capacitores de distinta medida.

El microcontrolador es programado por medio de la computadora, con la ayuda del software Atmel Studio y con el código, que se puede encontrar posteriormente en el documento. Este chip, al ser un microcontrolador, se encargará de llevar a cabo todo el procesamiento necesario para cumplir con el código.

En los ledes se muestra la hora. En las cuatro matrices podemos ver las unidades y decenas para las horas y para los minutos, mientras que en el *display* podemos observar los segundos, que se despliegan en formato binario.

También hicimos uso de cuatro MAX7219 que sirven para controlar las matrices de ledes, es decir, un chip por cada matriz.

Se implementaron cinco *switches*. Uno de ellos, sirve para determinar si la alarma está o no encendida. Los otros cuatro, sirven para seleccionar un número en binario; bien sea para ajustar la hora o configurar la alarma.

Para complementar la funcionalidad de los *switches*, se utilizaron cuatro botones. La idea es poner el número deseado con los switches y, después, presionar uno de los cuatro botones para registrar el número. Existe un botón que registrará el número en las unidades de minutos, otro en las decenas de minutos, otro para las unidades de horas y otro para las decenas de horas. El código cuenta con las validaciones necesarias para registrar solo horas existentes, es decir, no podrías configurar la hora 99:99, como máximo podría ser 23:59.

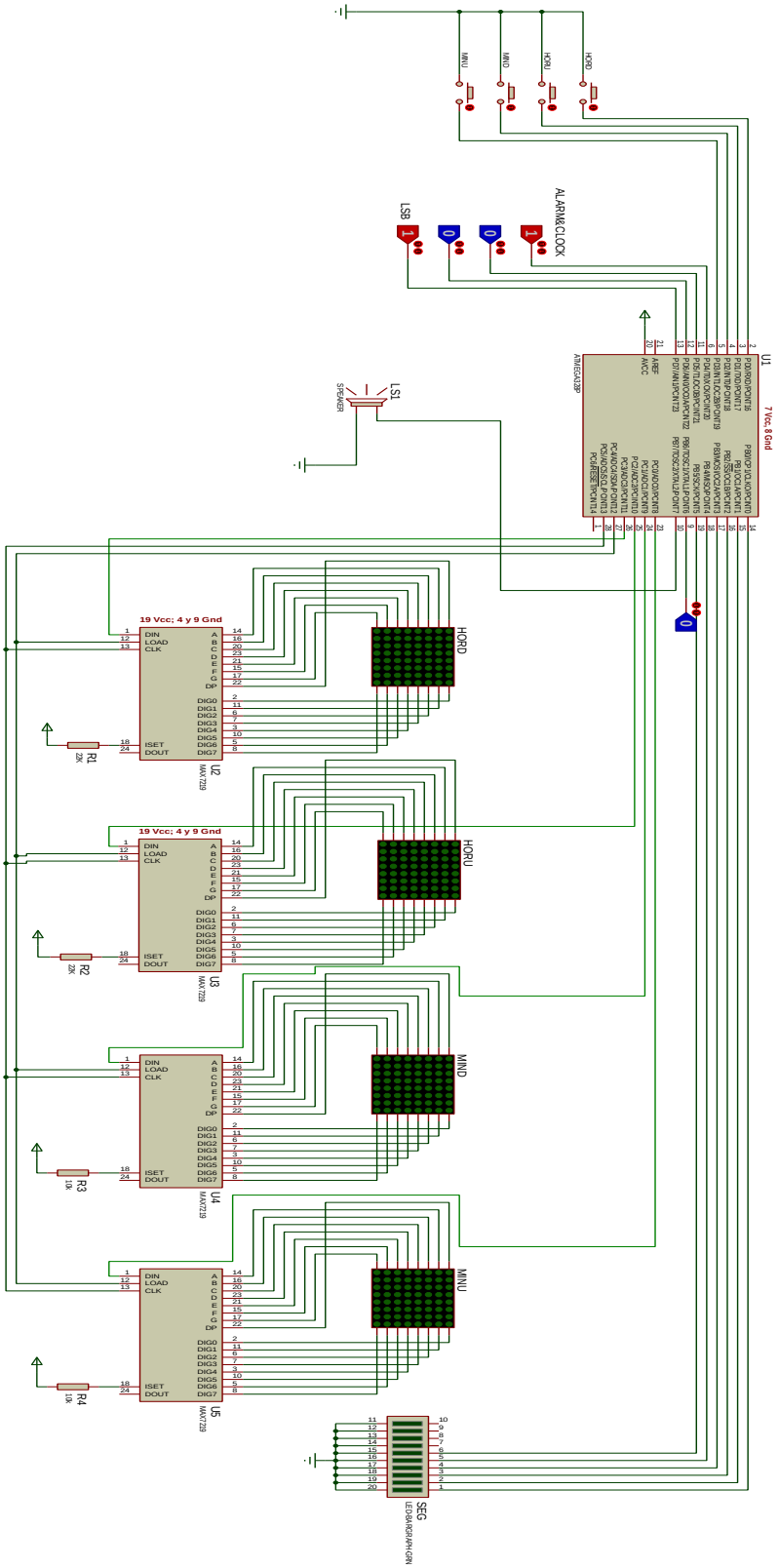
Finalmente, se utilizó una bocina, que sonará cuando la hora del reloj sea igual a la hora configurada por el usuario como alarma. Sonará durante un minuto.

Los capacitores fueron necesarios para eliminar ruido, la mayoría proveniente del uso de las matrices.

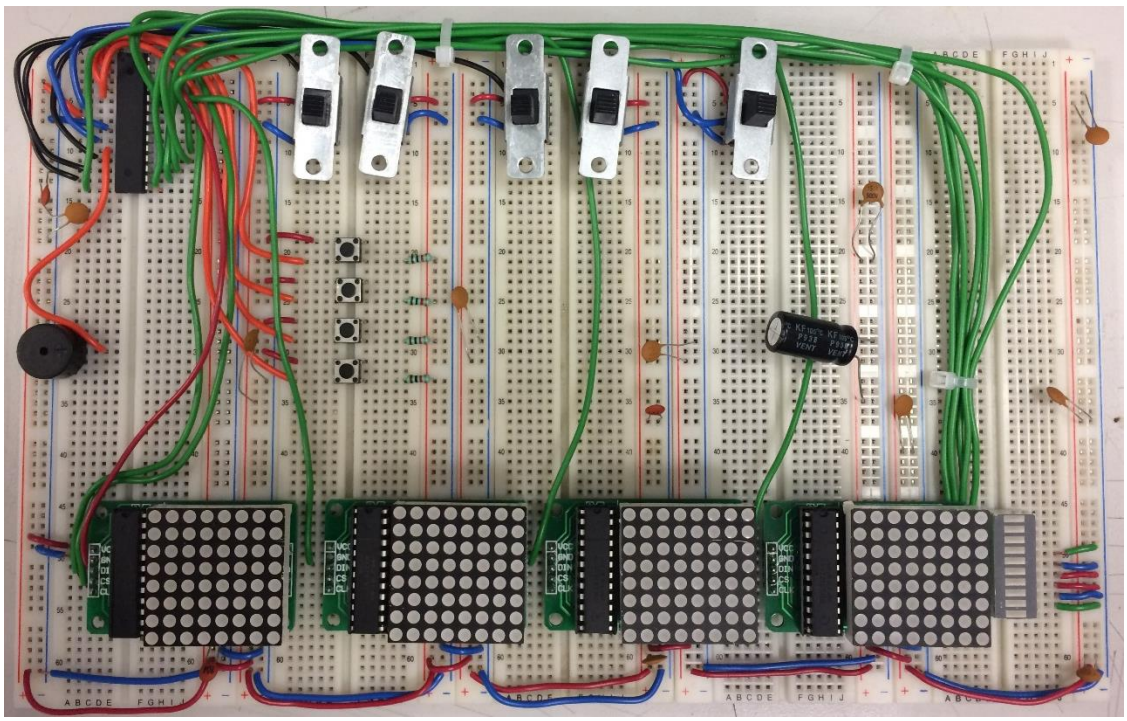
A continuación, incluimos una imagen del circuito en Proteus, fotografías del circuito físico, el código fuente necesario para programar el microcontrolador y nuestras conclusiones del proyecto. Nota: la imagen se exportó de Proteus con calidad, no obstante, por su tamaño no se aprecia completamente bien en este documento; por esta razón, incluimos un enlace de respaldo en Dropbox, que contiene todos los archivos que se necesitaron en la práctica, incluida la imagen del circuito para poder verla de una mejor manera. Enlace de respaldo:

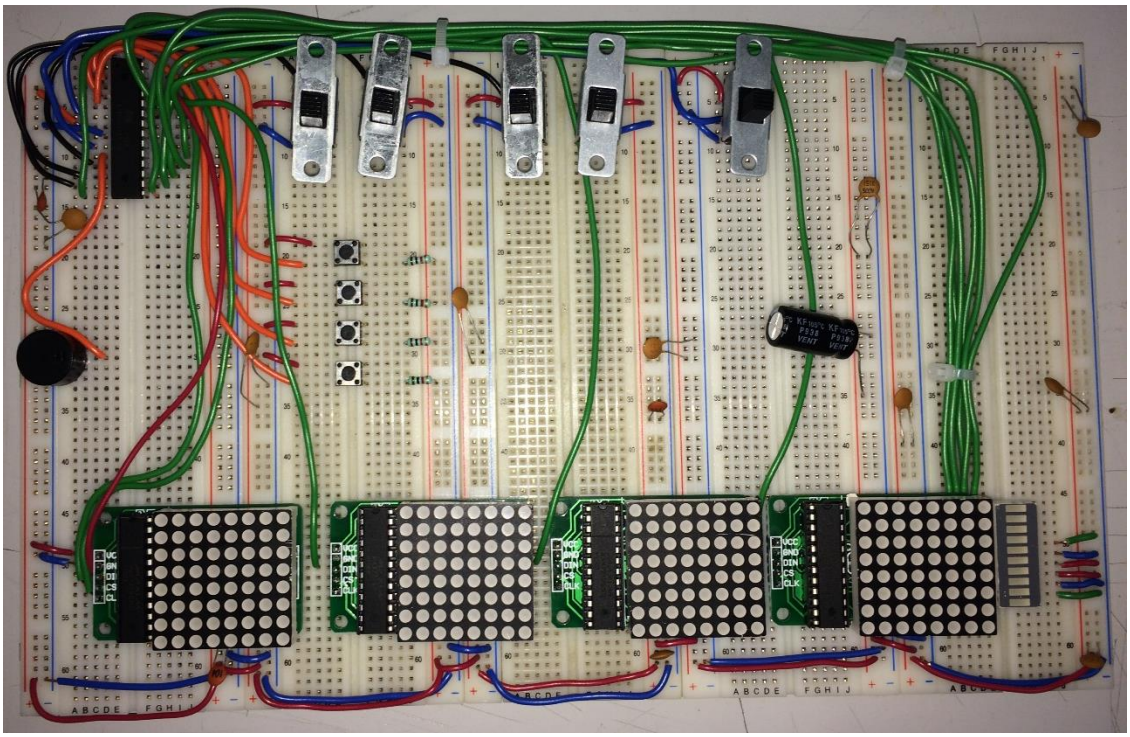
https://www.dropbox.com/sh/6eor19r5hvdc5hl/AAAg2xEx8nTXGKETDfKF2Hdsa?dl=0

Diagrama eléctrico completo de la práctica:



Fotografía del circuito:





Código fuente documentado (librería y código):

Librería:

```
//Librería para controlar la matriz de leds
 #include <delay.h>
 #define DIN_1  PORTC.0
 #define DIN_2  PORTC.1
 #define DIN_3  PORTC.2
 #define DIN_4  PORTC.3
 #define LOAD PORTC.4
 #define CLK  PORTC.5

//Control de forma individual de cada matriz
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;
}
void MandaMax7219_2(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_2=0;
      else
         DIN_2=1;
      CLK=1;
      CLK=0;
      dato=dato<<1; //esto es lo mismo a dividirlo entre 2, para recorrer datos
```

```
    }
    LOAD=1;
    LOAD=0;
}
void MandaMax7219_3(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_3=0;
        else
            DIN_3=1;
        CLK=1;
        CLK=0;
        dato=dato<<1; //esto es lo mismo a dividirlo entre 2, para recorrer datos
    }
    LOAD=1;
    LOAD=0;
}
void MandaMax7219_4(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_4=0;
        else
            DIN_4=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){
    DDRC|=0x31;
    MandaMax7219_1(0x0900);
    MandaMax7219_1(0x0A0B);
    MandaMax7219_1(0x0B07);
```

```
   MandaMax7219_1(0x0C01);
   MandaMax7219_1(0x0F00);
}
void ConfiguraMax_2(void){
   DDRC|=0x32;
   MandaMax7219_2(0x0900);
   MandaMax7219_2(0x0A0B);
   MandaMax7219_2(0x0B07);
   MandaMax7219_2(0x0C01);
   MandaMax7219_2(0x0F00);
}
void ConfiguraMax_3(void){
   DDRC|=0x34;
   MandaMax7219_3(0x0900);
   MandaMax7219_3(0x0A0B);
   MandaMax7219_3(0x0B07);
   MandaMax7219_3(0x0C01);
   MandaMax7219_3(0x0F00);
}
void ConfiguraMax_4(void){
   DDRC|=0x38;
   MandaMax7219_4(0x0900);
   MandaMax7219_4(0x0A0B);
   MandaMax7219_4(0x0B07);
   MandaMax7219_4(0x0C01);
   MandaMax7219_4(0x0F00);
}
//Numeros para las unidades de minuto
void MinUCrea_0(){
   MandaMax7219_1(0x0100);
   MandaMax7219_1(0x0200);
   MandaMax7219_1(0x033C);
   MandaMax7219_1(0x0442);
   MandaMax7219_1(0x0542);
   MandaMax7219_1(0x063C);
   MandaMax7219_1(0x0700);
   MandaMax7219_1(0x0800);
}
void MinUCrea_1(){
   MandaMax7219_1(0x0100);
   MandaMax7219_1(0x0200);
   MandaMax7219_1(0x0302);
   MandaMax7219_1(0x047E);
   MandaMax7219_1(0x0522);
   MandaMax7219_1(0x0600);
   MandaMax7219_1(0x0700);
```

```
    MandaMax7219_1(0x0800);
}
void MinUCrea_2(){
    MandaMax7219_1(0x0100);
    MandaMax7219_1(0x0200);
    MandaMax7219_1(0x0332);
    MandaMax7219_1(0x044A);
    MandaMax7219_1(0x0546);
    MandaMax7219_1(0x0622);
    MandaMax7219_1(0x0700);
    MandaMax7219_1(0x0800);
  }
void MinUCrea_3(){
    MandaMax7219_1(0x0100);
    MandaMax7219_1(0x0200);
    MandaMax7219_1(0x036C);
    MandaMax7219_1(0x0452);
    MandaMax7219_1(0x0542);
    MandaMax7219_1(0x0644);
    MandaMax7219_1(0x0700);
    MandaMax7219_1(0x0800);
    }
void MinUCrea_4(){
    MandaMax7219_1(0x0100);
    MandaMax7219_1(0x0200);
    MandaMax7219_1(0x037E);
    MandaMax7219_1(0x0410);
    MandaMax7219_1(0x0510);
    MandaMax7219_1(0x0670);
    MandaMax7219_1(0x0700);
    MandaMax7219_1(0x0800);
}
void MinUCrea_5(){
    MandaMax7219_1(0x0100);
    MandaMax7219_1(0x0200);
    MandaMax7219_1(0x034C);
    MandaMax7219_1(0x0452);
    MandaMax7219_1(0x0552);
    MandaMax7219_1(0x0672);
    MandaMax7219_1(0x0700);
    MandaMax7219_1(0x0800);
}
  void MinUCrea_6(){
    MandaMax7219_1(0x0100);
    MandaMax7219_1(0x0200);
    MandaMax7219_1(0x034C);
```

```
  MandaMax7219_1(0x0452);
  MandaMax7219_1(0x0552);
  MandaMax7219_1(0x063C);
  MandaMax7219_1(0x0700);
  MandaMax7219_1(0x0800);
}
void MinUCrea_7(){
  MandaMax7219_1(0x0100);
  MandaMax7219_1(0x0200);
  MandaMax7219_1(0x0370);
  MandaMax7219_1(0x044E);
  MandaMax7219_1(0x0540);
  MandaMax7219_1(0x0640);
  MandaMax7219_1(0x0700);
  MandaMax7219_1(0x0800);
}
void MinUCrea_8(){
  MandaMax7219_1(0x0100);
  MandaMax7219_1(0x0200);
  MandaMax7219_1(0x032C);
  MandaMax7219_1(0x0452);
  MandaMax7219_1(0x0552);
  MandaMax7219_1(0x062C);
  MandaMax7219_1(0x0700);
  MandaMax7219_1(0x0800);
}
void MinUCrea_9(){
  MandaMax7219_1(0x0100);
  MandaMax7219_1(0x0200);
  MandaMax7219_1(0x033C);
  MandaMax7219_1(0x044A);
  MandaMax7219_1(0x054A);
  MandaMax7219_1(0x0632);
  MandaMax7219_1(0x0700);
  MandaMax7219_1(0x0800);
}
//Numeros para las decenas de minuto
void MinDCrea_0(){
  MandaMax7219_2(0x0100);
  MandaMax7219_2(0x0200);
  MandaMax7219_2(0x033C);
  MandaMax7219_2(0x0442);
  MandaMax7219_2(0x0542);
  MandaMax7219_2(0x063C);
  MandaMax7219_2(0x0700);
  MandaMax7219_2(0x0800);
```

```
}
void MinDCrea_1(){
   MandaMax7219_2(0x0100);
   MandaMax7219_2(0x0200);
   MandaMax7219_2(0x0302);
   MandaMax7219_2(0x047E);
   MandaMax7219_2(0x0522);
   MandaMax7219_2(0x0600);
   MandaMax7219_2(0x0700);
   MandaMax7219_2(0x0800);
}
void MinDCrea_2(){
   MandaMax7219_2(0x0100);
   MandaMax7219_2(0x0200);
   MandaMax7219_2(0x0332);
   MandaMax7219_2(0x044A);
   MandaMax7219_2(0x0546);
   MandaMax7219_2(0x0622);
   MandaMax7219_2(0x0700);
   MandaMax7219_2(0x0800);
  }
void MinDCrea_3(){
   MandaMax7219_2(0x0100);
   MandaMax7219_2(0x0200);
   MandaMax7219_2(0x036C);
   MandaMax7219_2(0x0452);
   MandaMax7219_2(0x0542);
   MandaMax7219_2(0x0644);
   MandaMax7219_2(0x0700);
   MandaMax7219_2(0x0800);
   }
void MinDCrea_4(){
   MandaMax7219_2(0x0100);
   MandaMax7219_2(0x0200);
   MandaMax7219_2(0x037E);
   MandaMax7219_2(0x0410);
   MandaMax7219_2(0x0510);
   MandaMax7219_2(0x0670);
   MandaMax7219_2(0x0700);
   MandaMax7219_2(0x0800);
}
void MinDCrea_5(){
   MandaMax7219_2(0x0100);
   MandaMax7219_2(0x0200);
   MandaMax7219_2(0x034C);
   MandaMax7219_2(0x0452);
```

```
   MandaMax7219_2(0x0552);
   MandaMax7219_2(0x0672);
   MandaMax7219_2(0x0700);
   MandaMax7219_2(0x0800);
}
//Numeros para las unidades de hora
void HorUCrea_0(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x033C);
   MandaMax7219_3(0x0442);
   MandaMax7219_3(0x0542);
   MandaMax7219_3(0x063C);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_1(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x0302);
   MandaMax7219_3(0x047E);
   MandaMax7219_3(0x0522);
   MandaMax7219_3(0x0600);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_2(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x0332);
   MandaMax7219_3(0x044A);
   MandaMax7219_3(0x0546);
   MandaMax7219_3(0x0622);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
  }
void HorUCrea_3(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x036C);
   MandaMax7219_3(0x0452);
   MandaMax7219_3(0x0542);
   MandaMax7219_3(0x0644);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
   }
```

```
void HorUCrea_4(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x037E);
   MandaMax7219_3(0x0410);
   MandaMax7219_3(0x0510);
   MandaMax7219_3(0x0670);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_5(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x034C);
   MandaMax7219_3(0x0452);
   MandaMax7219_3(0x0552);
   MandaMax7219_3(0x0672);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_6(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x034C);
   MandaMax7219_3(0x0452);
   MandaMax7219_3(0x0552);
   MandaMax7219_3(0x063C);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_7(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x0370);
   MandaMax7219_3(0x044E);
   MandaMax7219_3(0x0540);
   MandaMax7219_3(0x0640);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_8(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x032C);
   MandaMax7219_3(0x0452);
   MandaMax7219_3(0x0552);
```

```
   MandaMax7219_3(0x062C);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
void HorUCrea_9(){
   MandaMax7219_3(0x0100);
   MandaMax7219_3(0x0200);
   MandaMax7219_3(0x033C);
   MandaMax7219_3(0x044A);
   MandaMax7219_3(0x054A);
   MandaMax7219_3(0x0632);
   MandaMax7219_3(0x0700);
   MandaMax7219_3(0x0800);
}
//Numeros para las decenas de hora
void HorDCrea_0(){
   MandaMax7219_4(0x0100);
   MandaMax7219_4(0x0200);
   MandaMax7219_4(0x033C);
   MandaMax7219_4(0x0442);
   MandaMax7219_4(0x0542);
   MandaMax7219_4(0x063C);
   MandaMax7219_4(0x0700);
   MandaMax7219_4(0x0800);
}
void HorDCrea_1(){
   MandaMax7219_4(0x0100);
   MandaMax7219_4(0x0200);
   MandaMax7219_4(0x0302);
   MandaMax7219_4(0x047E);
   MandaMax7219_4(0x0522);
   MandaMax7219_4(0x0600);
   MandaMax7219_4(0x0700);
   MandaMax7219_4(0x0800);
}
void HorDCrea_2(){
   MandaMax7219_4(0x0100);
   MandaMax7219_4(0x0200);
   MandaMax7219_4(0x0332);
   MandaMax7219_4(0x044A);
   MandaMax7219_4(0x0546);
   MandaMax7219_4(0x0622);
   MandaMax7219_4(0x0700);
   MandaMax7219_4(0x0800);
  }
```

Código:

```
#include <mega328P.h>
#include <delay.h>
#include "matriz.h"
#include <io.h>

unsigned char seg=0;
unsigned int minU=0,minD=0,hrU=0,hrD=0;
unsigned int BMINU,BMIND,BHORU,BHORD;
unsigned int Ent;
unsigned int ALHD=1,ALHU=1,ALMD=1,ALMU=1,SAlar,Bsleep;
unsigned long int i=0;
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
   PORTB=seg;
   seg++;
   if(seg>59){
   seg=0;
   minU++;
      if(minU>9){
         minD++;
         minU=0;
         i=0;
         if(minD>5){
            hrU++;
            minD=0;
            if(hrU>9){
               hrD++;
               hrU=0;

            }
         }
      }

   }
   if(hrU==4 && hrD==2){
      minU=0;
      minD=0;
      hrU=0;
      hrD=0;
   }
   if(ALHD==hrD && ALHU==hrU && ALMD==minD && ALMU==minU){
      PORTB.7=673;
```

```
        }

}

void main(void){
   ConfiguraMax_1();
   ConfiguraMax_2();
   ConfiguraMax_3();
   ConfiguraMax_4();
   DDRB=0xBF;
   PORTD=0xFF;
   DDRC.6=1;
   PORTB.6=1;
   //    DDRD=0xFF;  //PD0 a PD7 de salida
   // Timer/Counter 1 initialization
   // Clock source: System Clock
   // Clock value: 15.625 kHz
   // Mode: CTC top=OCR1A
   // OC1A output: Discon.
   // OC1B output: Discon.
   // Noise Canceler: Off
   // Input Capture on Falling Edge
   // Timer1 Overflow Interrupt: Off
   // Input Capture Interrupt: Off
   // Compare A Match Interrupt: On
   // Compare B Match Interrupt: Off
   TCCR1A=0x00;
   TCCR1B=0x0B;

   OCR1AH=15624/256;
   OCR1AL=15624%256;

   // Timer/Counter 1 Interrupt(s) initialization
   TIMSK1=0x02;

   // Global enable interrupts
   #asm("sei")
   while(1)
   {
      BHORD=PIND.0;
      BHORU=PIND.1;
      BMIND=PIND.2;
      BMINU=PIND.3;
      SAlar=PINB.6;
      Bsleep=PINB.7;
      Ent=PIND>>4;
```

```
    if(minU==0){
       MinUCrea_0();
    }
    if(minU==1){
       MinUCrea_1();
    }
    if(minU==2){
       MinUCrea_2();
    }
    if(minU==3){
       MinUCrea_3();
    }
    if(minU==4){
       MinUCrea_4();
    }
    if(minU==5){
       MinUCrea_5();
    }
    if(minU==6){
       MinUCrea_6();
    }
    if(minU==7){
       MinUCrea_7();
    }
    if(minU==8){
       MinUCrea_8();
    }
    if(minU==9){
       MinUCrea_9();
    }
    if(minD==0){
       MinDCrea_0();
    }
    if(minD==1){
       MinDCrea_1();
    }
    if(minD==2){
       MinDCrea_2();
    }
    if(minD==3){
       MinDCrea_3();
    }
    if(minD==4){
       MinDCrea_4();
    }
    if(minD==5){
```

```
        MinDCrea_5();
    }
    if(hrU==0){
        HorUCrea_0();
    }
    if(hrU==1){
        HorUCrea_1();
    }
    if(hrU==2){
        HorUCrea_2();
    }
    if(hrU==3){
        HorUCrea_3();
    }
    if(hrU==4){
        HorUCrea_4();
    }
    if(hrU==5){
        HorUCrea_5();
    }
    if(hrU==6){
        HorUCrea_6();
    }
    if(hrU==7){
        HorUCrea_7();
    }
    if(hrU==8){
        HorUCrea_8();
    }
    if(hrU==9){
        HorUCrea_9();
    }
    if(hrD==0){
        HorDCrea_0();
    }
    if(hrD==1){
        HorDCrea_1();
    }
    if(hrD==2){
        HorDCrea_2();
    }
    if(SAlar==1){
        if(BMINU==0){
        ALMU=Ent;
    }
    if(BMIND==0){
```

```
        ALMD=Ent;
    }
    if(BHORU==0){
        if(ALHD!=2){
            ALHU=Ent;
        }
        else if(ALHD==2){
            if(Ent<4){
                ALHU=Ent;
            }
        }
    }
    if(BHORD==0){
        if(ALHU>4){
        if(Ent==1 || Ent==0){
            ALHD=Ent;
        }
        else{
             ALHD=ALHD;
            }
        }
        else{
             ALHD=Ent;
        }

    }

    }
    else{
    if(BMINU==0){
        minU=Ent;
    }
    if(BMIND==0){
        minD=Ent;
    }
    if(BHORU==0){
        if(hrD!=2){
            hrU=Ent;
        }
        else if(hrD==2){
            if(Ent<4){
                hrU=Ent;
            }
        }
    }
    if(BHORD==0){
```

```
            if(hrU>4){
            if(Ent==1 || Ent==0){
                hrD=Ent;
            }
            else{
                 hrD=hrD;
                }
            }
            else{
                 hrD=Ent;
            }

        }
        }

    }

}
```

Conclusiones:

Gerardo Naranjo:

Esta práctica final me fue de utilidad para reafirmar el uso de las interrupciones; en nuestro caso, se requieren para generar un retardo de un segundo exacto y que es más preciso que la función "*delay*" y menos complicado que utilizar varios "NOP". En cuanto al código, fue relativamente sencillo el hacer las validaciones requeridas por un reloj. El pintar en la matriz el número deseado también fue sencillo, gracias al uso del MAX7219. En conclusión, resultó ser un código extenso (con todo y librería), pero que es bastante simple y el circuito no conlleva gran complejidad; por lo tanto, un reloj con alarma fácil de hacer con un micro.

Roberto Figueroa:

En nuestro proyecto final pusimos en práctica alguno de los conocimientos adquiridos en la materia de microcontroladores, utilizamos las interrupciones para llevar una cuenta de los segundos del reloj, poniéndole condiciones para que cambiara en los minutos y posteriormente en las horas, mostramos estos en los *displays* de leds, los cuales se pintaban por columnas cada número. Fue relativamente sencillo realizar todo esto, aunque quisimos ponerle más cosas a nuestro reloj, tuvimos un inconveniente, ya que usamos todos los pines, lo que nos limitó a la hora de desarrollarlo más a fondo.

Alejandro Ossio:

En el proyecto final pusimos en práctica los conocimientos adquiridos durante el semestre, principalmente las interrupciones, ya que gracias a estas pudimos obtener un reloj más preciso que si hubiéramos utilizado simples *delays*, también vimos pudimos observar que con las interrupciones somos capaces de hacer más cosas y no nos limitamos con el programa principal. Después de haber cursado esta materia considero que salgo con una herramienta más a utilizar para futuros proyectos.

Referencias:

ATMEL. (s.f.). *Datasheet ATMega328P* . Obtenido de 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash.