Laboratorio de microcontroladores.
Alejando Ossio Díaz, A01209122.
Gerardo Daniel Naranjo Gallegos, A01209499.
Roberto Figueroa Saavedra, A01209689.
09 de septiembre de 2016.
Práctica #2.

Reloj binario.

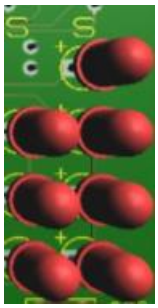
Descripción:

En esta práctica realizaremos un reloj que mostrará la hora en ledes, con un formato hh/mm/ss en BCD. Será controlado mediante un microcontrolador ATMega8, tendrá una batería para su alimentación, cuatro *push-botton para ajuste del reloj* y un *speaker*, entre otros.

El código para el reloj será programado en el microcontrolador, quien se encargará de ejecutarlo. El reloj muestra la hora en un formato de 24 horas. También, contará con una alarma o sonido que se activa cada media hora y dura un minuto o hasta que se oprima algún botón, lo que ocurra primero. El código se muestra más adelante.

Los *push-botton* sirven para ajustar las horas y minutos del reloj. Cada uno aumenta en uno, bien sea los minutos o las horas. Un tercer botón reinicia los segundos a cero.

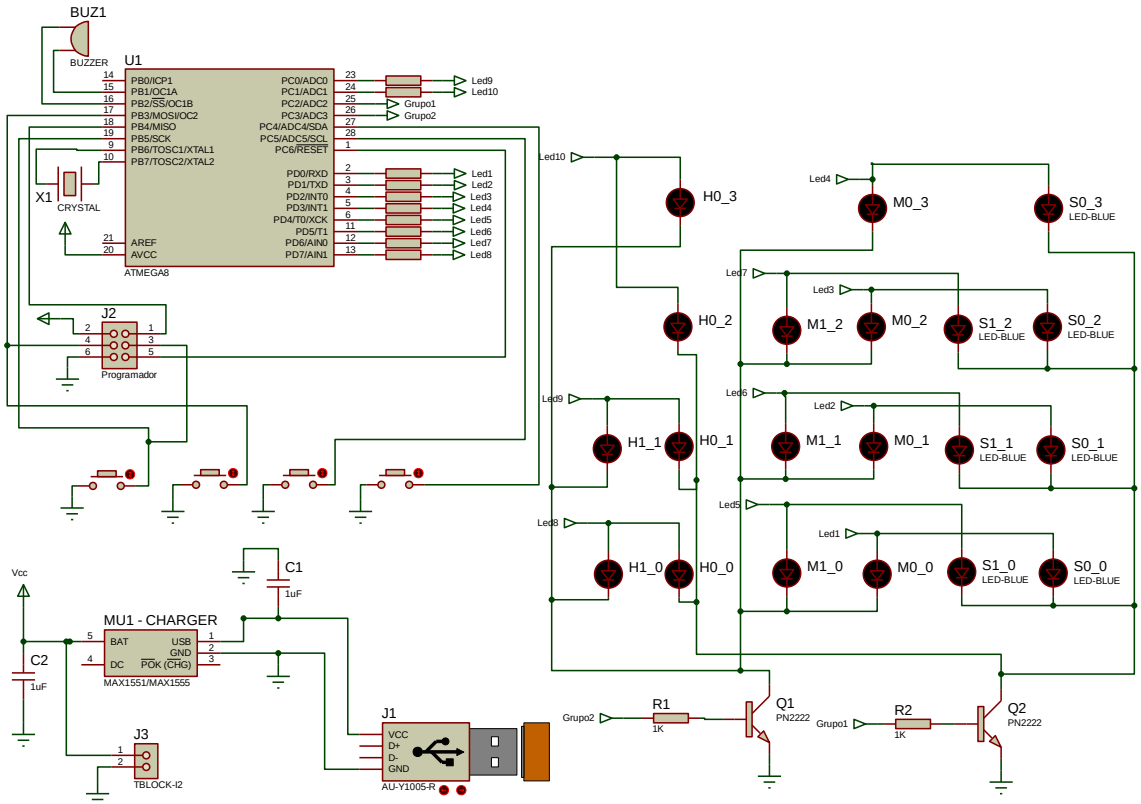
Los ledes nos indicarán las horas. Recordemos que se muestra en binario… es decir, tomando por ejemplo a los segundos, la primera columna de izquierda a derecha representa las decenas y la siguiente columna las unidades; en donde el primer led, de abajo hacia arriba, es un uno, el segundo vale dos, el tercero vale cuatro y el cuarto led vale ocho unidades. En decenas la misma lógica.



El *speaker* emite un tono, a manera de alarma, cada media hora.

La batería sirve para alimentar al circuito en general (ledes, microcontrolador, speaker, etc).

Diagrama eléctrico:



Código fuente:

El código fuente utilizado en el microcontrolador es el siguiente:

```
#include <io.h>
   #include <mega8.h>
   #include <delay.h>
   #define Buzzer1 PORTB.1
   #define Buzzer2 PORTB.2
   #define led1 PORTD.0
   #define led2 PORTD.1
   #define led3 PORTD.2
   #define led4 PORTD.3
   #define led5 PORTD.4
   #define led6 PORTD.5
   #define led7 PORTD.6
   #define led8 PORTD.7
   #define led9 PORTC.0
   #define led10 PORTC.1
   #define grupo1 PORTC.2
   #define grupo2 PORTC.3
```

```
   unsigned int segU=0,segD=0,minU=0,minD=0,hrU=0,hrD=0;
```

```
   unsigned char x=0,y=0,i=0,j=0;
   bit GPO=1;
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
//timer cuentas del reloj en unidades y decenas
   {
      segU++;
      if(segU>9){
         segU=0;
         segD++;
         if(segD>5){
            segD=0;
            minU++;
            if(minU>9){
               minD++;
               minU=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;
      }

   }

// Timer2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
   GPO=!GPO;
   //asiganciones a 0 para evitar espejismo en los led's al hacer refresh de
grupos
         led1=0;
         led2=0;
         led3=0;
         led4=0;
         led5=0;
         led6=0;
         led7=0;
         led8=0;
```

```
            led9=0;
            led10=0;
            //control grrupo 1
        if(GPO==0){
            grupo1=1;
            grupo2=0;
            led1=(segU)&1;
            led2=(segU>>1)&1;
            led3=(segU>>2)&1;
            led4=(segU>>3)&1;
            led5=(segD)&1;
            led6=(segD>>1)&1;
            led7=(segD>>2)&1;
            led8=(hrU)&1;
            led9=(hrU>>1)&1;
            led10=(hrU>>2)&1;
        }
        //control grupo 2
           else{
            grupo2=1;
            grupo1=0;
            led1=(minU)&1;
            led2=(minU>>1)&1;
            led3=(minU>>2)&1;
            led4=(minU>>3)&1;
            led5=(minD)&1;
            led6=(minD>>1)&1;
            led7=(minD>>2)&1;
            led8=(hrD)&1;
            led9=(hrD>>1)&1;
            led10=(hrU>>3)&1;


        }

}

//silbido
   #pragma warn-
   void delay_US(unsigned char x)
   {
       #asm
       ld r26,y
       Ciclo:
       NOP
       NOP
       NOP
       NOP
       NOP
       DEC R26
       BRNE Ciclo
```

```
      #endasm
   }
      #pragma warn+
   void silbido()
   {
      unsigned char x,y;
      for (x = 250;x > 70;x--)
      for (y = 0;y < 3 ;y++)
      {
      Buzzer2=1;
      Buzzer1=0;
      delay_US(x);
      Buzzer2=0;
      Buzzer1=1;
      delay_US(x);
      }
      Buzzer2=0;
      Buzzer1=0;
   }
void main(){
   DDRD=0xFF;
   DDRC=0x0F;
   DDRB=0x06;
   PORTB.3=1;
   PORTC=0x30;
   TCCR1A=0x00;
   TCCR1B=0x0C;
   OCR1AH=125000/64;
   OCR1AL=125000%64;
   TIMSK=0x90;
   // Timer/Counter 2 initialization
   // Clock source: System Clock
   // Clock value: 62.500 kHz
   // Mode: CTC top=OCR2
   // OC2 output: Disconnected
   ASSR=0x00;
   TCCR2=0x0C;
   TCNT2=0x00;
   //OCR2=62499;
   OCR2=250;
   // Global enable interrupts
   #asm("sei")

   while(1){
     //alarma cada 30 min
    if(((minD==3 && minU==0) || (minD==0 && minU==0)) ){
      silbido();
    }
    //reset segundos
    if(PINC.4==0){
    segU=0;
```

```
    segD=0;
    }
    //aumentar minutos
    if(PINC.5==0){
     x=1;
    }
    if(PINC.5==1 && x==1){
          y=1;
    }
    if(x==1 && y==1){
      minU++;
      if(minU==10){
          minD++;
          minU=0;
      }
      if(minD==6){
      minD=0;
      hrU++;
      }
      x=0;
      y=0;
      }
      //aumentar horas
    if(PINB.3==0){
     i=1;
    }
    if(PINB.3==1 && i==1){
          j=1;
    }
    if(i==1 && j==1){
      hrU++;
      if(hrU==10){
          hrD++;
          hrU=0;
      }
      if(hrD==2&& hrU==4){
          hrU=0;
          hrD=0;
      }
      i=0;
      j=0;
      }
    }

  }
```

Fotografía del circuito:

Conclusiones individuales:

Alejandro Ossio:

Con esta práctica sencilla se pudo repasando acerca de lo visto en micro controladores, específicamente el manejo de *timers* y sus registros para poder así realizar trabajos más complejos sin necesidad de detenerse para recordar lo esencial del micro controlador. También nos permitió practicar en nuestras habilidades para soldar.

Gerardo Naranjo:

El desarrollo del hardware en esta práctica fue algo diferente, pues en lugar de usar los clásicos protoboards, se elaboró en una placa, lo que da mayor presentación al proyecto y a un costo modesto. Permitió la práctica al soldar. En cuanto al software, el realizar el código ayudó sobre todo a recordar las interrupciones, vistas en el curso anterior, de microcontroladores.

Roberto Figueroa:

Con esta práctica recordamos el uso de interrupciones con los *timers*, para poder realizar varias cosas con el micro, sin necesidad de detenerse cada que quiera ejecutar una acción y tuvimos que armar nosotros mismos el hardware que utilizamos, desarrollando así nuestra practica de soldar.