Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

#LAB. CONTROLADORES Y MICROCONTROLADORES PROGRAMABLES

#Práctica P5
" Contador con interrupciones por timer "

*Nombre o nombres de los integrantes junto con su matrícula:

#Verónica Yazmín Gómez Cruz #1884224
#Nahaliel Gamaliel Ríos Martínez #1884244

#Ing. Jesús Daniel Garza Camarena

Semestre Febrero 2021 – Junio 2021

MN1N2

San Nicolás de los Garza, N.L. #15.04.2021

Objetivo

Investigar sobre el uso de los TIMER en los microcontroladores

Introducción.

Un Timer es un módulo interno de un microcontrolador. Además el timer puede generar una señal periódica a una frecuencia que puede ser configurada. La función principal del timer es contar automáticamente a la velocidad de su frecuencia configurada. Por ejemplo, sus principales características son los bits que puede «contar».

El timer-0 del ATMEGA328P es de 8 bits, si se configura a una frecuencia de 100Hz, esto es un periodo de T = 1/100Hz = 10mS, le llevaría contar, automáticamente de 0 a 255 (8-bit) un tiempo de 255*10mS = 2.55s. Cuando en ATMEGA328P, el Timer termina su cuenta, esté genera una interrupción. Esto indicaría que ha la cuenta ha terminado. Dicha interrupción puede avisar o notificar al procesador para ejecutar alguna función específica. Por ejemplo, nos permite medir el tiempo transcurrido sin el uso de retardos de tiempo que llegan a detener el funcionamiento del procesador.

En el ATMEGA328P, así como en todos los microcontroladores, los timers funcionan de forma independiente al procesador. Por lo tanto, podemos hacer otros procesos mientras esperamos a que el módulo nos avise que ha terminado su cuenta. Esto nos permite usar, de forma más eficiente, el tiempo del microprocesador del ATMEGA328P.

En el ATMEGA328P, existen 3 Timers internos. El primer timer, el timer-0 es de 8 bits, el Timer-1 es de 16 bits y el Timer-2 es de 8 bits. Es decir, que pueden contar de 0 a 255 (8-bit) o de 0 a 65535 (16 bits).

Diagrama de bloques

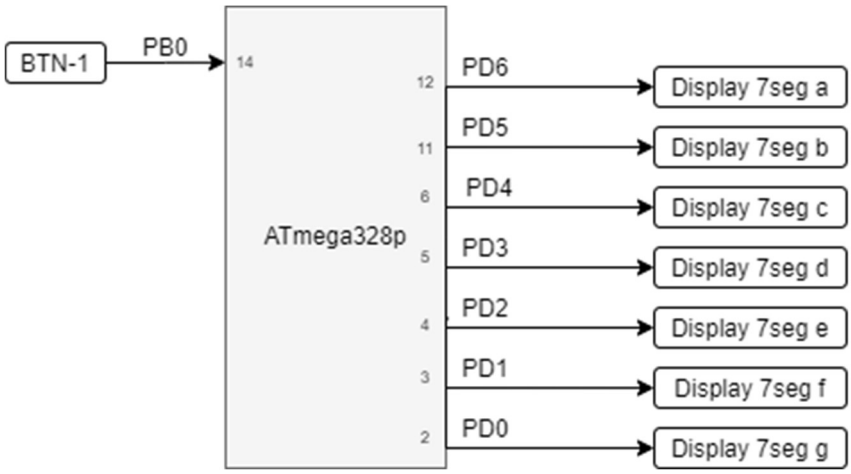
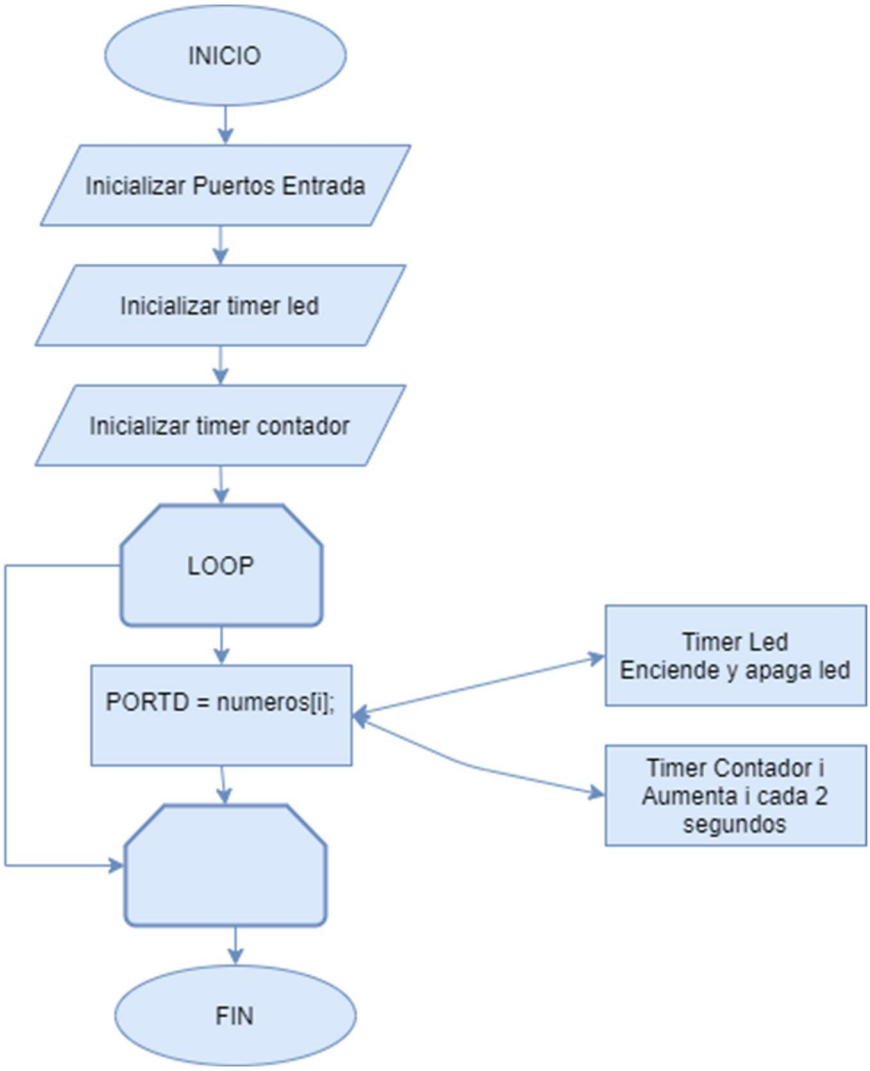


Diagrama de flujo.



Materiales utilizados

1 ATMEGA328P
1 Display 7 seg cc
1 Led
9 Resistencia 220
Jumpers de distintos colores

Código en Atmel.

```
/*************************************************
 * LLENAR ESTE ESPACIO CON LOS SIGUIENTES DATOS:    *
 * Nombre: Verónica Yazmín Gómez Cruz               *
 *         Nahaliel Gamaliel Rios Martinez          *
```

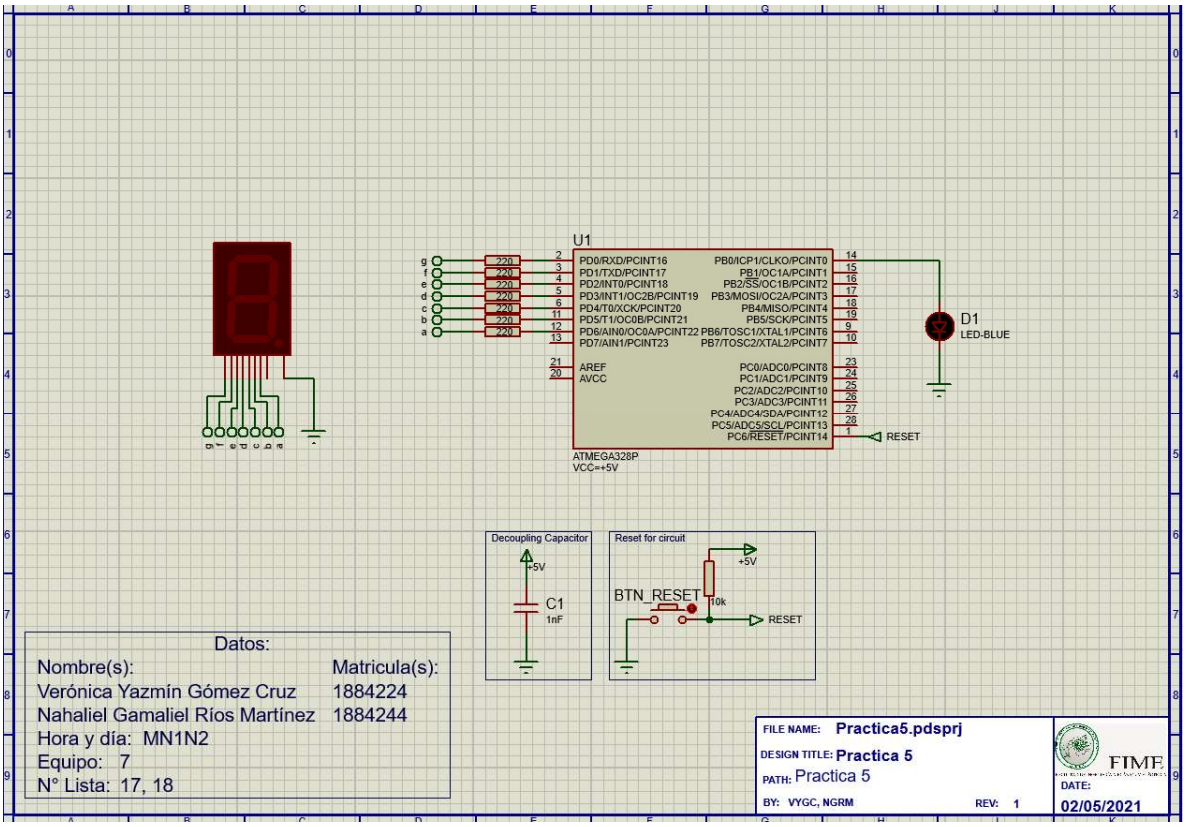
```
 * Hora clase: N1-N2                                  *
 * Día: M                                             *
 * N° de lista: 17, 18                                *
 * N° de Equipo: 7                                    *
 * Dispositivo: ATMEGA328P                            *
 * Rev: 1.0                                           *
 * Propósito de la actividad:                         *
 * Diseñar un contador ascendente de 0 a 9 a partir *
 * de la práctica anterior, el incremento sera por  *
 * medio de la interrupción por el timer del        *
 * microcontrolador no se debe de utilizar la       *
 * bilbioteca delay , el tiempo para incrementar lo *
 * define el alumno, además se debe de agregar un   *
 * led que parpadee a 0.1 segundo de velocidad para *
 * demostrar que el código se está ejecutando.      *
 *                                Fecha: 02.05.2021 *
 ************************************************/
/*atmega328P PIN - OUT*/
/*       PIN - OUT
         atmega328P
          -------
    PC6  |1     28| PC5
    PD0  |2     27| PC4
    PD1  |3     26| PC3
    PD2  |4     25| PC2
    PD3  |5     24| PC1
    PD4  |6     23| PC0
    VCC  |7     22| GND
    GND  |8     21| AREF
    PB6  |9     20| AVCC
    PB7  |10    19| PB5
    PD5  |11    18| PB4
    PD6  |12    17| PB3
    PD7  |13    16| PB2
    PB0  |14    15| PB1
          --------
*/
/*atmega328P PIN FUNCTIONS*/
 /*
 atmega328P PIN FUNCTIONS
 pin   function              name    pin     function              name
 1     !RESET/PCINT14        PC6     15      PCINT1/OC1A           PB1
 2     RxD/PCINT16           PD0     16      PCINT2/OC1B/SS        PB2
 3     TxD/PCINT17           PD1     17      PCINT3/OC2A/MOSI      PB3
 4     INT0/PCINT18          PD2     18      PCINT4/MISO           PB4
 5     INT1/PCINT19/OC2B     PD3     19      PCINT5/SCK            PB5
 6     PCINT20               PD4     20      ANALOG VCC            AVCC
 7     +5v                   VCC     21      ANALOG REFERENCE      AREF
 8     GND                   GND     22      GND                   GND
 9     XTAL1/PCINT6          PB6     23      PCINT8/ADC0           PC0
 10    XTAL2/PCINT7          PB7     24      PCINT9/ADC1           PC1
 11    PCINT21/OC0B          PD5     25      PCINT10/ADC2          PC2
 12    PCINT22/OC0A/AIN0     PD6     26      PCINT11/ADC3          PC3
 13    PCINT23/AIN1          PD7     27      PCINT12/ADC4/SDA      PC4
 14    PCINT0/AIN1           PB0     28      PCINT13/ADC5/SCL      PC5
 */
/******************Bibliotecas*****************************/
#include <avr/io.h>//se incluyen las Bibliotecas de E/S del AVR atmega328P
```

```
#include <avr/interrupt.h>// librería de interrupciones

/*****************Macros y constantes*********************/
#define F_CPU 1000000UL //1 Mhz

/******************Variables globales*********************/
//--Espacio para declarar variables globales
#define a PIND0
#define b PIND1
#define c PIND2
#define d PIND3
#define e PIND4
#define f PIND5
#define g PIND6
#define LedIndicador PINB0

volatile char i = 0; //Contador para leer el arreglo de numeros
volatile char timer = 0; //Contador para el timer

uint8_t numeros[10] = {
	//gfedcba
	0b0111111, //0
	0b0000110, //1
	0b1011011, //2
	0b1001111, //3
	0b1100110, //4
	0b1101101, //5
	0b1111101, //6
	0b1000111, //7
	0b1111111, //8
	0b1100111, //9
};


/***********************Funciones*************************/
//--Espacio para Establecer funciones
/*********************Declaración de Funciones************/
//--Espacio para declarar funciones
void initialize_ports(void);
void initialize_timer_led(void);// Función para inicializar Timer_0
void initialize_timer_i(void);// Función para inicializar Timer_1
void timer_led_on(void);// Función para encender Timer_0
void timer_i_on(void);// Función para apagar Timer_0

/******************Programa principal*********************/
int main(void)
{
//--Inicialización
    cli(); //Deshabilitamos interrupciones
	initialize_ports(); // va hacía la inicialización de puertos
	initialize_timer_led();// va hacía la inicialización del TIMER para controlar
Led
	initialize_timer_i();// va hacía la inicialización del TIMER para controlar
Contador del display
	sei(); //Habilitamos interrupciones
	timer_led_on(); //Encendemos Timer0
	timer_i_on(); //Encendemos Timer0
```

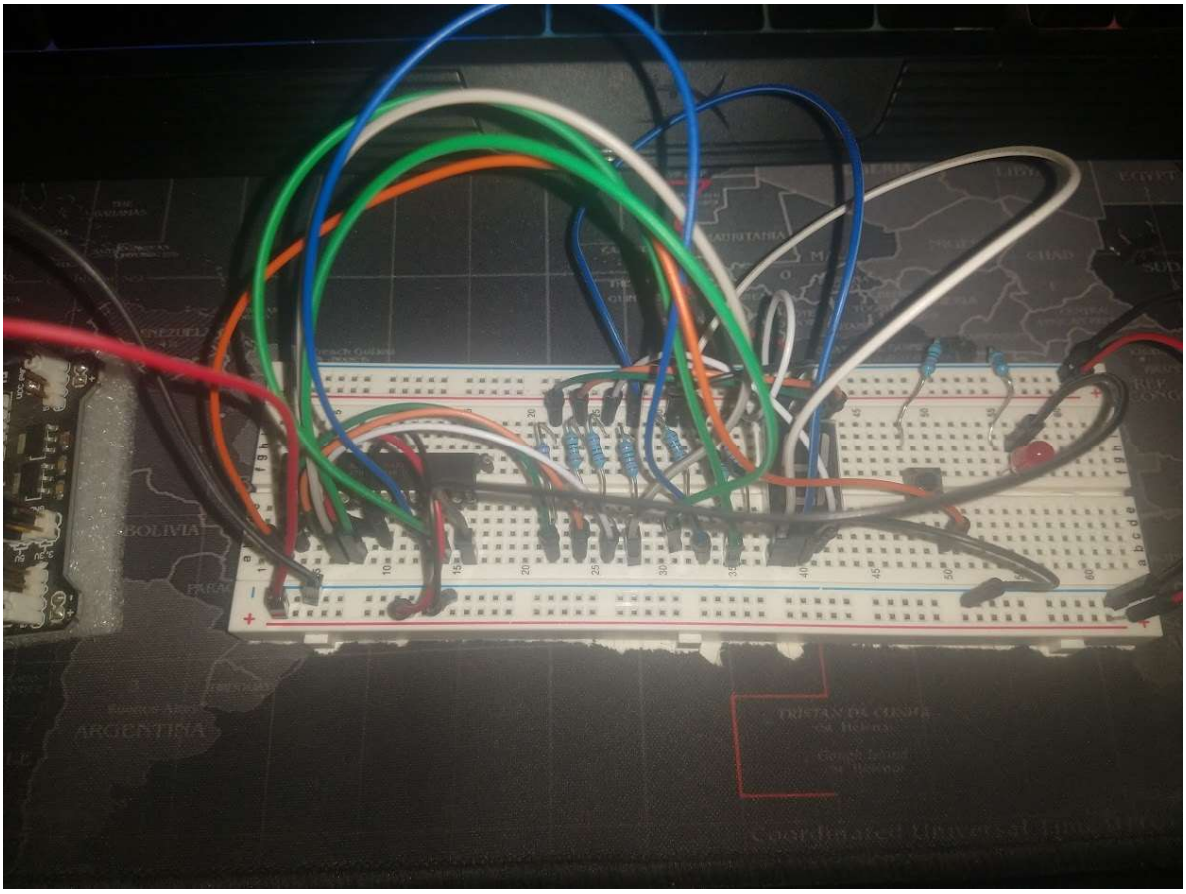
```
//--Ejecución
    while (1) //loop infinito
    {

              //PORTB |=_BV(LedIndicador); //Encender
              PORTD = numeros[i];

              if (i == 10) {
                     i = 0;
              }

    } // END loop infinito

} // END MAIN
/*******************Definición de funciones*****************/
//*************************************************************
//Descripcion de lo que hace la funcion:                     *
//initialize_ports : inicializa los puertos de entrada o     *
//salida                                                     *
//*************************************************************
void initialize_ports(void)
{

    //--Salidas
       DDRD |=_BV(a);
       DDRD |=_BV(b);
       DDRD |=_BV(c);
       DDRD |=_BV(d);
       DDRD |=_BV(e);
       DDRD |=_BV(f);
       DDRD |=_BV(g);

       DDRB |=_BV(LedIndicador);

       PORTB = 0x00; //-Por seguridad iniciamos en 0
       PORTD = 0x00; //-Por seguridad iniciamos en 0

}
//*********************************************************************
//initialize_timer_led : inicializa el timer para controlar Led
//*********************************************************************
void initialize_timer_led(void)
{
    //Modo de operacíon configurado como CTC
    TCCR0A &=~ (1<<WGM00);// 0 en el bit WGM00
    TCCR0A |=  (1<<WGM01);// 1 en el bit WGM01
    TCCR0B &=~ (1<<WGM02);// 0 en el bit WGM02
    OCR0A = 97; //Registro de 8 bits donde se pone el numero a comparar
    TIMSK0 |= (1<<OCIE0A);//Se pone un 1 en el bit OCIE0A del registro
    //TIMSK0 para habilitar la interrupción
}
//*********************************************************************
//timer_led_on: Enciende el timer para controlar Led
//*********************************************************************
void timer_led_on(void)
{

    TCNT0 = 0; // Registro de 8 bits que lleva el conteo del timer_0
```

```
    //Prescaler configurado en 1024
    TCCR0B |=  (1<<CS00);// 1 en el bit CS00
    TCCR0B &=~ (1<<CS01);// 0 en el bit CS01
    TCCR0B |=  (1<<CS02);// 1 en el bit CS02

}
//**********************************************************************
//initialize_timer1 : inicializa el timer para el contador del display
//**********************************************************************
void initialize_timer_i(void)
{
//Modo de operacíon configurado como CTC
    TCCR1A &=~ _BV(WGM10);// 0 en el bit WGM10
    TCCR1A &=~ _BV(WGM11); // 0 en el bit WGM11
    TCCR1B |=  _BV(WGM12); // 1 en el bit WGM12
    TCCR1B &=~ _BV(WGM13);// 0 en el bit WGM13
    OCR1A = 488; //Registro de 8 bits donde se pone el numero a comparar
    TIMSK1 |= (1<<OCIE1A); //Se pone un 1 en el bit OCIE1A del registro
}
//**********************************************************************
//timer_i_on: inicializa el timer para el contador del display
//**********************************************************************
void timer_i_on(void)
{
       TCNT1 = 0;// Registro de 16 bits que lleva el conteo del timer_0
       //Prescaler configurado en 1024
       TCCR1B |=  (1<<CS10);// 1 en el bit CS10
       TCCR1B &=~ (1<<CS11);// 0 en el bit CS11
       TCCR1B |=  (1<<CS12);// 1 en el bit CS12

}
//**********************************************************************
//Vectores de interrupción, Interrupt service routine (ISR)
//**********************************************************************
//**********************************************************************
//TIMER_0
//**********************************************************************
ISR (TIMER0_COMPA_vect)// Vector de interrupción para el Timer0 (0.1s)
{
    PORTB ^= (1<<LedIndicador); //Encender y apagar led indicador
}
//**********************************************************************
//TIMER_1
//**********************************************************************
ISR (TIMER1_COMPA_vect)// Vector de interrupción para el Timer0 (0.5s)
{
       timer++;
       if(timer == 2){ //Cada 2 Seg Aumenta
              i++;
              timer = 0;
       }
}
```

Diagrama del circuito en PROTEUS.



Fotografía del Protoboard armado



Conclusión

En esta práctica trabajamos con los timers que nos proporciona el atmega328P para poder controlar 2 eventos por separado, el contador que nos iba a permitir que el display de 7 segmentos mostrara todos los números del 1 al 9 que teníamos almacenados en nuestro arreglo y otro que nos permitía encender y apagar un led cada 0.1 segundos. Considero que las aplicaciones que pueden llegar a tener este tipo de eventos pueden ser muy útiles en cualquier tipo de proyecto, por ejemplo, si queremos automatizar algún dispensador de comida que se ejecute cada x tiempo, o cualquier proceso donde el tiempo sea un factor para considerar, esto se vuelve también muy útil porque nuestro código principal puede seguir ejecutándose hasta tener la respuesta del timer, a diferencia del delay que detiene todo el código principal hasta que termina el tiempo programado.

Bibliografía

A. (2017, 29 diciembre). Arduino timer - Interrupciones con el Timer2. HETPRO/TUTORIALES. https://hetpro-store.com/TUTORIALES/arduino-timer/

ATmega328P. 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. DATASHEET. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf