



# INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y  
TECNOLOGÍAS AVANZADAS

## “Práctica 2: Manejo de interrupción timer”

Asignatura: Microprocesadores, microcontroladores e interfaz

Docente: Trejo Salazar David Benjamín

Equipo 2:

- Aguilar Zarazúa Luis Fernando
- Martínez Pérez Nestor



Grupo: 2MM6  
Fecha de entrega: 10/04/2016

# ÍNDICE

1.-INTRODUCCIÓN .....	1
2.- OBJETIVOS.....	1
2.1.- GENERAL .....	1
2.2.- PARTICULARES.....	1
3.- MARCO TEÓRICO .....	1
3.1.- TEMPORIZADOR.....	1
3.1.1.- TCCR (TIMER COUNTER CONTROL REGISTER).....	1
3.1.2.- TCNT (TIMER COUNTER REGISTER) .....	1
3.1.3.- TIMSK (TIMER INTERRUPT MASK REGISTER) .....	2
4.- MATERIAL .....	2
5.- DESARROLLO EXPERIMENTAL .....	2
5.1.- CONTROL DE UN MOTOR A PASOS.....	2
5.1.1.- DIAGRAMA DE FLUJO .....	2
5.1.2.- DIAGRAMA ELÉCTRICO .....	7
5.1.3.- CÁLCULOS .....	8
5.1.4.- PROCEDIMIENTO.....	8
5.2.- CONTROL DE UN LED RGB.....	11
5.2.1.- DIAGRAMA DE FLUJO .....	11
5.2.2.- DIAGRAMA ELÉCTRICO .....	18
5.2.3.-CÁLCULOS .....	19
5.2.4.- PROCEDIMIENTO.....	19
5.3.- GENERADOR DE FRECUENCIAS .....	21
5.3.1.- DIAGRAMA DE FLUJO .....	21
5.3.2.- DIAGRAMA ELÉCTRICO .....	25
5.2.3.-CÁLCULOS .....	26
5.3.4.- PROCEDIMIENTO.....	27
6.- RESULTADOS .....	28
6.1.- CONTROL DE UN MOTOR A PASOS.....	28
6.2.- CONTROL DE UN LED RGB.....	29
6.3.- GENERADOR DE FRECUENCIAS .....	31
7.- CONCLUSIONES.....	33
- Aguilar Zarazúa Luis Fernando .....	33
- Martínez Pérez Nestor .....	33
8.- REFERENCIAS DE INTERNET .....	33

## ÍNDICE DE FIGURAS

<b>Figura 1.-</b> Circuito del ejercicio 1 a realizar en el laboratorio. ....	8
<b>Figura 2.-</b> Circuito correspondiente al primer ejercicio de la práctica 2. ....	9
<b>Figura 3.-</b> Segmento del código escrito en Atmel Studio.....	10
<b>Figura 4.-</b> Localización de la opción construir. ....	10
<b>Figura 5.-</b> Selección del archivo .hex para simularlo en proteus.....	10
<b>Figura 6.-</b> Circuito a realizar en el laboratorio correspondiente al segundo ejercicio.....	18
<b>Figura 7.-</b> Circuito correspondiente al segundo ejercicio de la práctica 2. ....	20
<b>Figura 8.-</b> Segmento de código del segundo ejercicio. ....	21
<b>Figura 9.-</b> Circuito a realizar en el laboratorio correspondiente al tercer ejercicio.....	26
<b>Figura 10.-</b> Frecuencias a reproducir. ....	27
<b>Figura 11.-</b> Circuito correspondiente al tercer ejercicio de la práctica 2. ....	27
<b>Figura 12.-</b> Segmento de código del tercer ejercicio.....	28
<b>Figura 13.-</b> Simulación en proteus para el primer ejercicio.....	29
<b>Figura 14.-</b> Prueba del ejercicio 1. ....	29
<b>Figura 15.-</b> Simulación en proteus para el segundo ejercicio. ....	30
<b>Figura 16.-</b> Prueba 1 del ejercicio 2, color azul. ....	30
<b>Figura 17.-</b> Prueba 2 del ejercicio 2, color verde.....	31
<b>Figura 18.-</b> Prueba 3 del ejercicio 2, color rojo. ....	31
<b>Figura 19.-</b> Simulación en proteus para el tercer ejercicio.....	32
<b>Figura 20.-</b> Prueba del ejercicio 3. ....	32

## **1.-INTRODUCCIÓN**

En esta práctica se realizaron tres ejercicios de los cuales el primero de ellos tiene la finalidad de crear el control para un motor a pasos para que este pueda girar en un sentido horario o en sentido anti horario usando para esto DIP switches, el motor debe de tener una frecuencia de paso de 240 Hz.

En el segundo ejercicio se creó un control para poder aumentar o disminuir la luminosidad de las tres capas de color de un led RGB.

Y finalmente para el ejercicio 3 se realizó un programa capaz de enviar a las salidas del microcontrolador pulsos con diferentes frecuencias para generar las notas musicales do, re, mi, fa, sol, la y si y sus correspondientes sostenidos de las notas que poseen.

## **2.- OBJETIVOS**

### **2.1.- GENERAL**

1. Que el alumno conozca el uso de interrupciones (timer) para poder obtener diferentes voltajes y frecuencias a la salida de los datos del microcontrolador ATmega328P.

### **2.2.- PARTICULARES**

1. Programar correctamente las instrucciones.
2. Armar correctamente los circuitos eléctricos necesarios.

## **3.- MARCO TEÓRICO**

### **3.1.- TEMPORIZADOR**

Los temporizadores son características estándar de casi todos los microcontroladores. Un microcontrolador AVR tiene temporizadores muy poderosos y multifuncionales.

Un temporizador de expresión más simple es un registro. Los temporizadores generalmente tienen una resolución de 8 o 16 bits. Así que un temporizador de 8 bits va de 0 a 255. Su valor aumenta / disminuye de forma automática a una velocidad predefinida (suministrado por el usuario) y esta operación no necesita la intervención de la CPU.

#### **3.1.1.- TCCR (TIMER COUNTER CONTROL REGISTER)**

Es utilizado para configurar el temporizador.

Hay 8 bits en este registro, cada uno utilizado para determinados fines, los últimos 3 bits CS02, CS01, CS00 que son los bits de selección de reloj.

#### **3.1.2.- TCNT (TIMER COUNTER REGISTER)**

Es la base de todos los modos de temporizador. Cuenta con el reloj del sistema, con el reloj preescalado o con un reloj externo.

### 3.1.3.- TIMSK (TIMER INTERRUPT MASK REGISTER)

Este registro se utiliza para activar y desactivar las interrupciones relacionadas con los temporizadores.

## 4.- MATERIAL

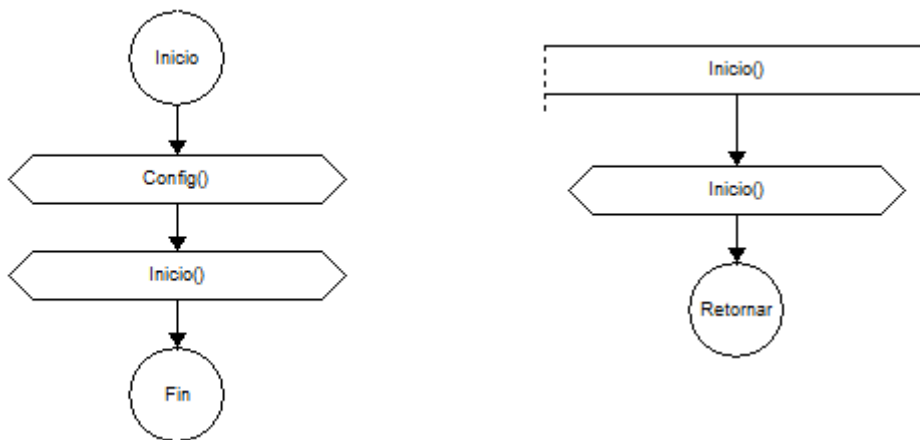
1. Fuente de alimentación.
2. ATmega328P.
3. Circuito eléctrico.
4. Programador.
5. PC.

## 5.- DESARROLLO EXPERIMENTAL

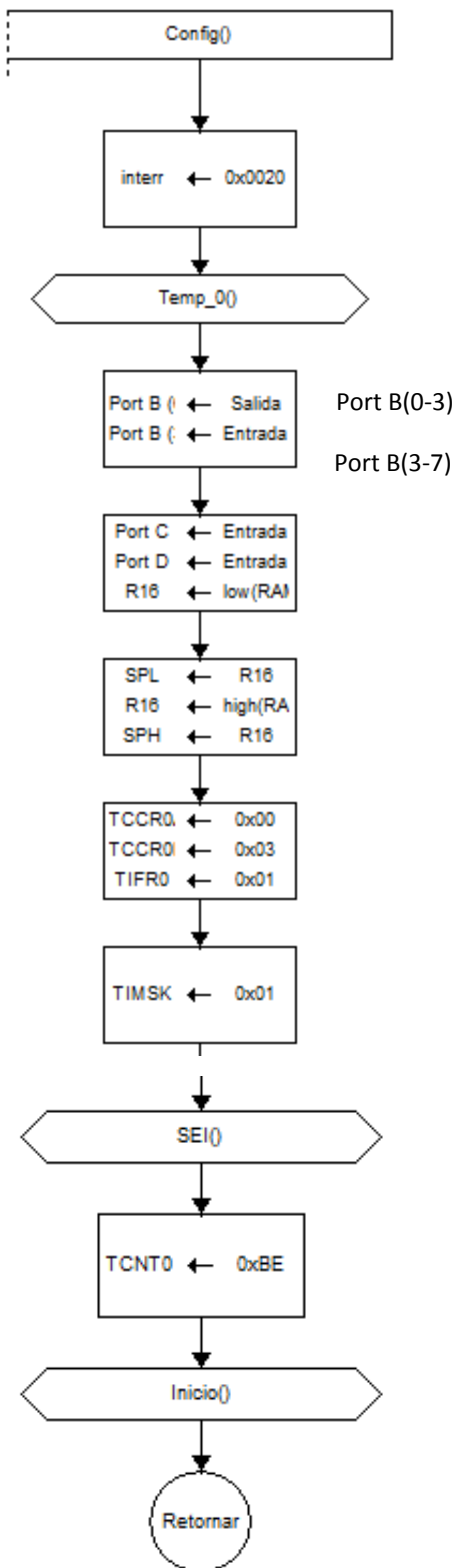
### 5.1.- CONTROL DE UN MOTOR A PASOS

#### 5.1.1.- DIAGRAMA DE FLUJO

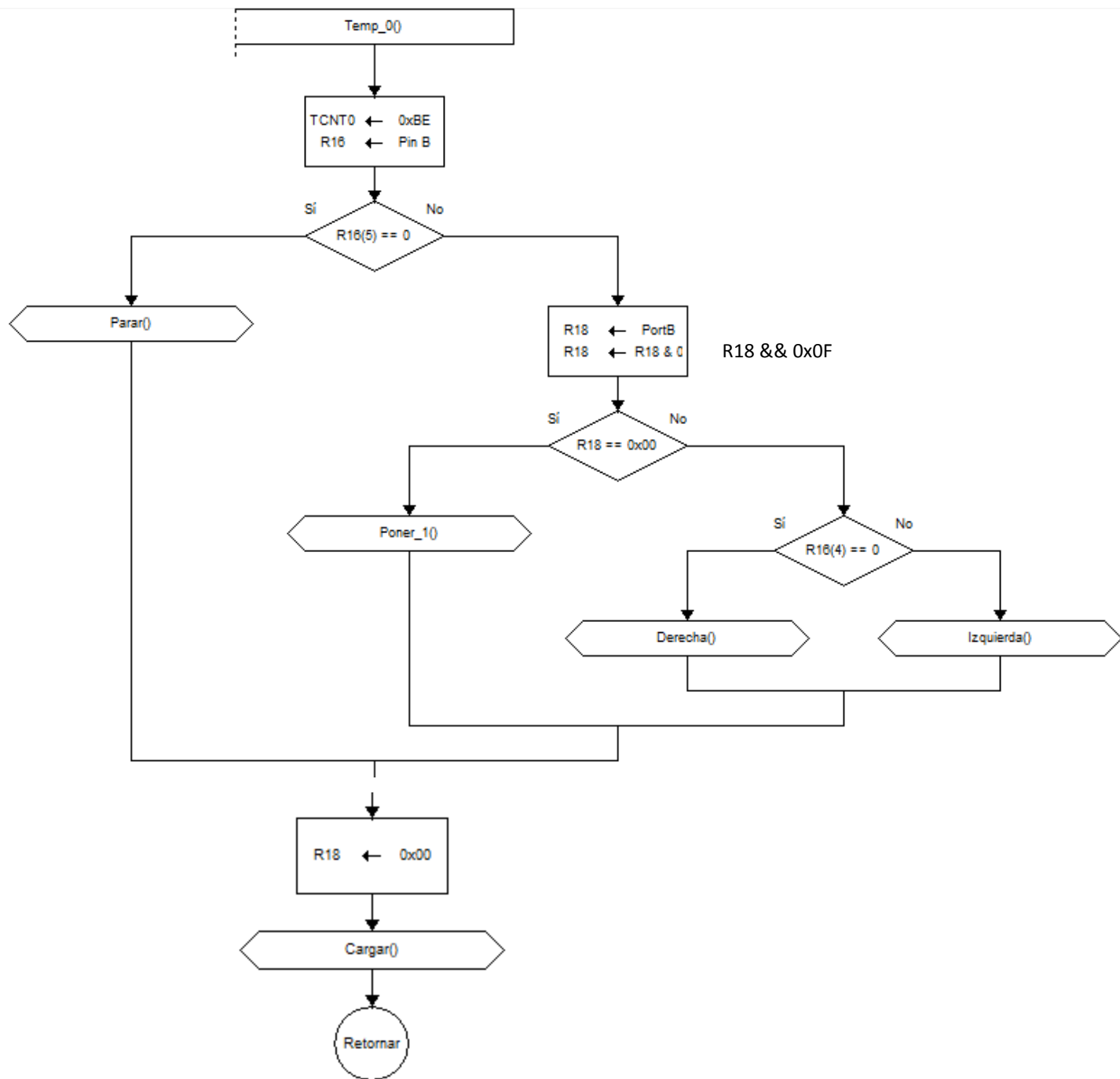
A continuación se presenta el diagrama de flujo correspondiente al primer ejercicio de la práctica dos. También se coloca la subrutina inicio.



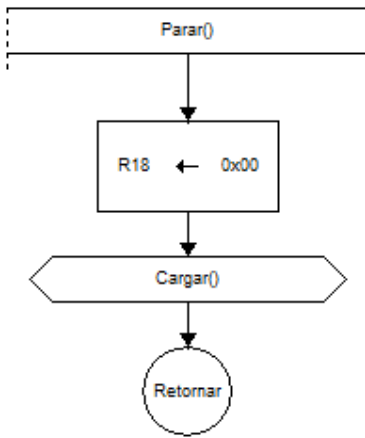
A continuación se presenta la subrutina config.



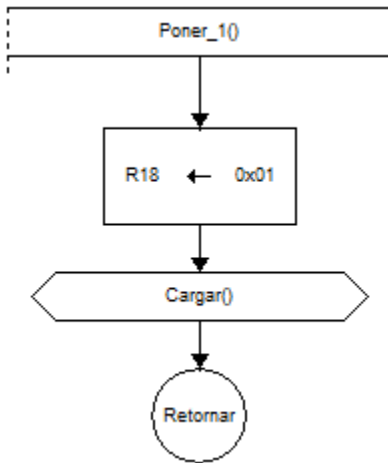
A continuación se presenta la subrutina Temp\_0.



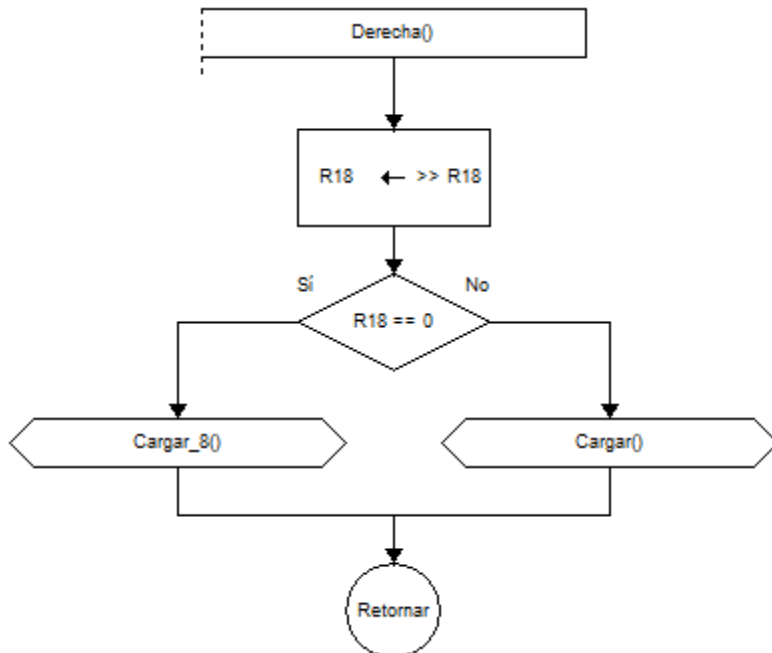
A continuación se presenta la subrutina Parar.



A continuación se presenta la subrutina `Poner_1`.

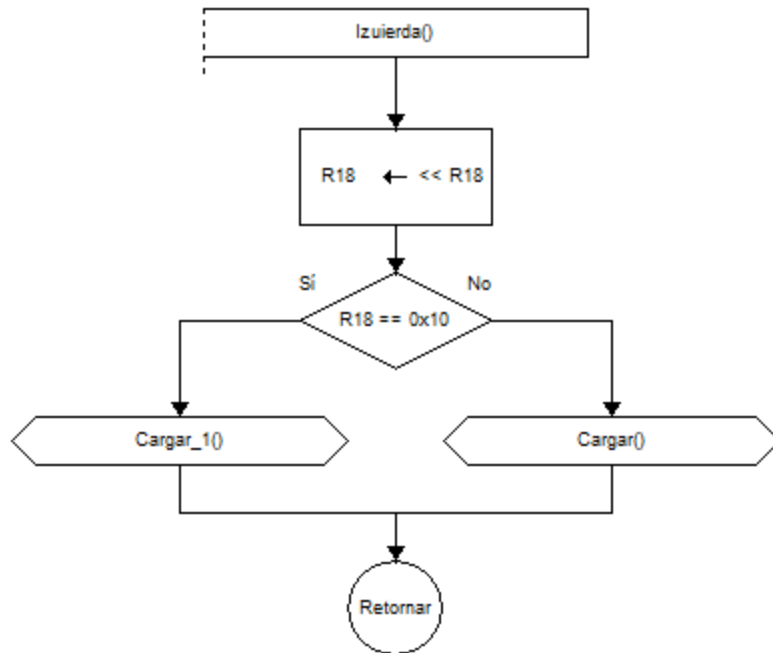


A continuación se presenta la subrutina `Derecha`.

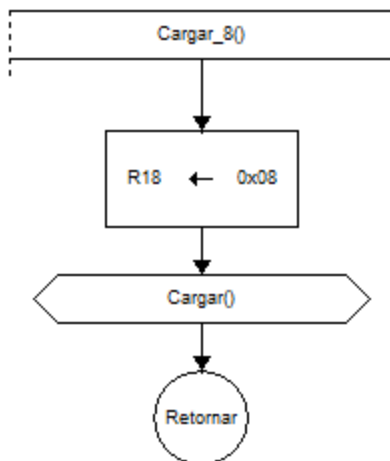




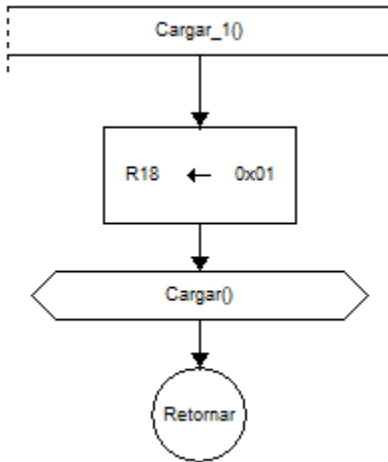
A continuación se presenta la subrutina Izquierda.



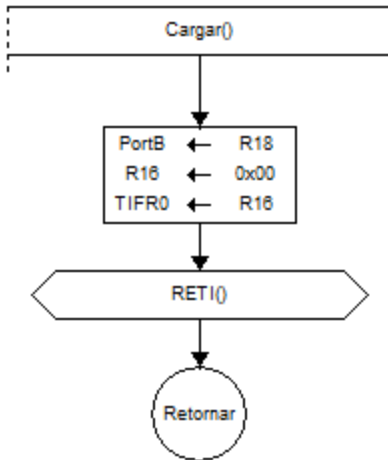
A continuación se presenta la subrutina Cargar\_8.



A continuación se presenta la subrutina Cargar\_1.



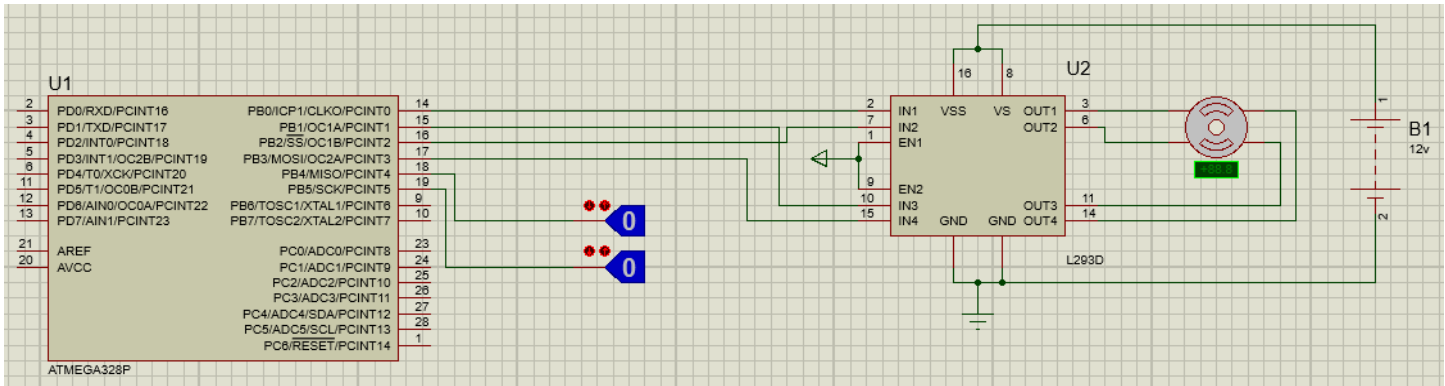
A continuación se presenta la subrutina Cargar.



Esas fueron todas las rutinas y subrutinas correspondientes a este ejercicio de la práctica 2 pero debido al software utilizado, algunos datos no quedaron visibles por lo que se colocaron cuadros de texto que aclararan los valores y operaciones que no estuvieran visibles.

### 5.1.2.- DIAGRAMA ELÉCTRICO

A continuación se presenta el diagrama del circuito a realizar en el laboratorio.



**Figura 1.-** Circuito del ejercicio 1 a realizar en el laboratorio.

Como se observa en la **figura 1** se utilizaron en la simulación probadores lógicos para simplificar el circuito simulado pero en la práctica se conectaron DIP switches para observar el comportamiento de este primer ejercicio.

### 5.1.3.- CÁLCULOS

$$Fp = \frac{1 * 10^6}{64 * (256 - Carga\ del\ Timer)} = 240Hz$$

$$\frac{1}{256 - Carga\ del\ Timer} = \frac{64 * 240}{1 * 10^6}$$

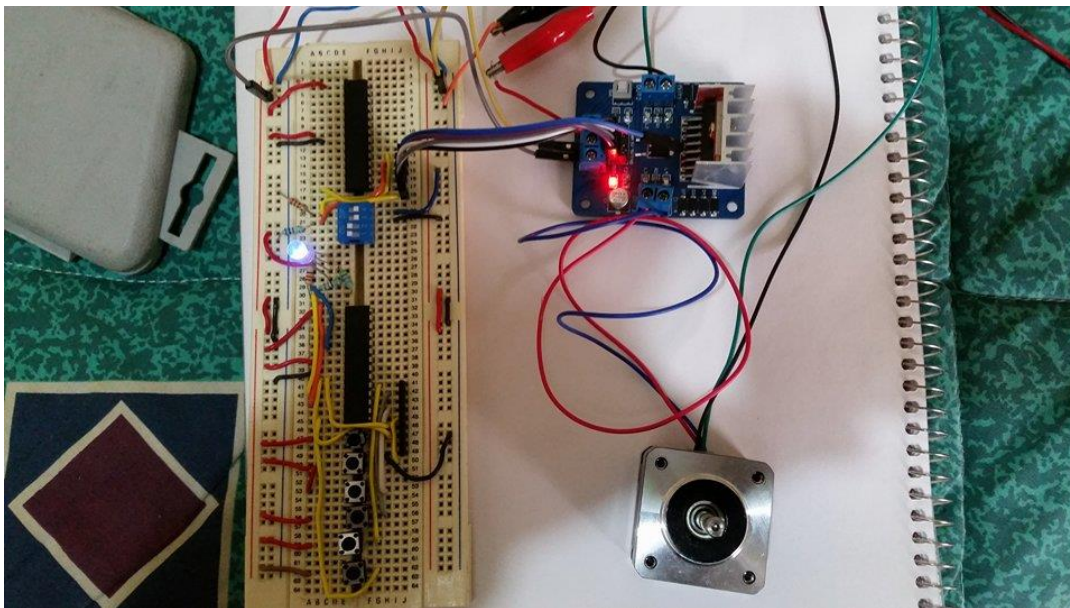
$$Carga\ del\ Timer = 256 - \frac{1 * 10^6}{64 * 240} \approx 190.89583\ Hz \approx 191\ Hz$$

$$Fp = \frac{1 * 10^6}{64 * (256 - 191)} = \frac{3125}{13} \approx 240.3846\ Hz$$

### 5.1.4.- PROCEDIMIENTO

1. Realizar un programa que controle un motor a pasos con la ayuda de la interrupción timer overflow (calcular una frecuencia de paso de 240 Hz), el control debe de considerar el encendido y apagado y tener en cuenta que no se debe energizar ninguna bobina cuando esté apagado; en el control también se debe considerar el cambio de giro (a favor y en contra de las manecillas del reloj).

Para este ejercicio se realizó el circuito que a continuación se muestra:



**Figura 2.-** Circuito correspondiente al primer ejercicio de la práctica 2.

Una vez armado el circuito de la **figura 2** se procedió a crear un nuevo proyecto en el software Atmel Studio en el cual se escribió el código en ensamblador correspondiente al primer ejercicio de esta práctica.

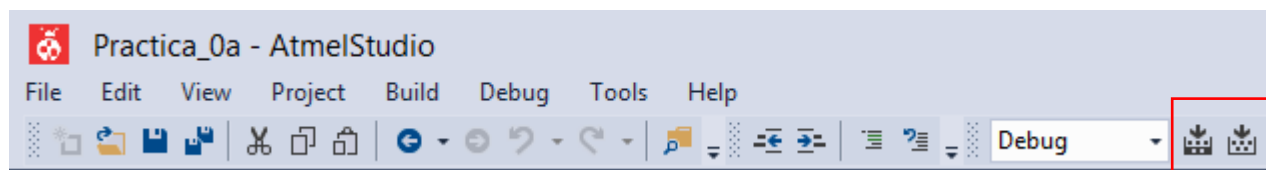
```

Practica_2A.asm - AtmelStudio
File Edit View Project Debug Tools Help
Practica_2A.asm
.include "m328pdef.inc"; Archivo de definicion de etiquetas.
.org 0x3C00 jmp config; Manda al inicio si esta configurado como bootloader.
.org 0x0000 jmp config; Manda al inicio.
.org 0x0020 jmp Temp_0; Ejecuta interrupción por Timer 0.
config:
    ldi R16, 0x0F; Limpia el registro 16.
    out DDRB, R16; Configura el puerto B del 0 al 3 como salida y los demas como entrada
    clr R16;
    out DDRC, R16; Configura el puerto C como entrada.
    out DDRD, R16; Configura el puerto D como entrada.
    ldi R16, low(RAMEND); Carga el nivel bajo de la pila.
    out SPL, R16; Carga el Stack pointer low con la direccion correcta.
    ldi R16, high(RAMEND); Carga el nivel alto de la pila.
    out SPH, R16; Carga el Stack pointer high con la direccion correcta.
    clr R16; Limpia R16.
    out TCCR0A, R16; Limpia el Registro TCCR0A para dejarlo en modo Normal.
    ldi R16, 0x03; Carga 0x03 a R16.
    out TCCR0B, R16; Usar el prescalador a un factor de 64.
    ldi R16, 0x01; Habilita TOV0.
    out TIFR0, R16; Limpia las interrupciones pendientes TOV0
    ldi R16, 0x01; Carga 0x01 a R16.
    sts TIMSK0, R16; Habilita la interrupción de TIMER0 por overflow.
    sei; Habilita interrupciones globales.
    clr R17;
    ldi R16, 0xBE; Limpia R16.
    out TCNT0, R16; Carga inicial del timer de 190 para lograr la frecuencia de 240Hz
    jmp inicio; F=240.3846Hz.

```

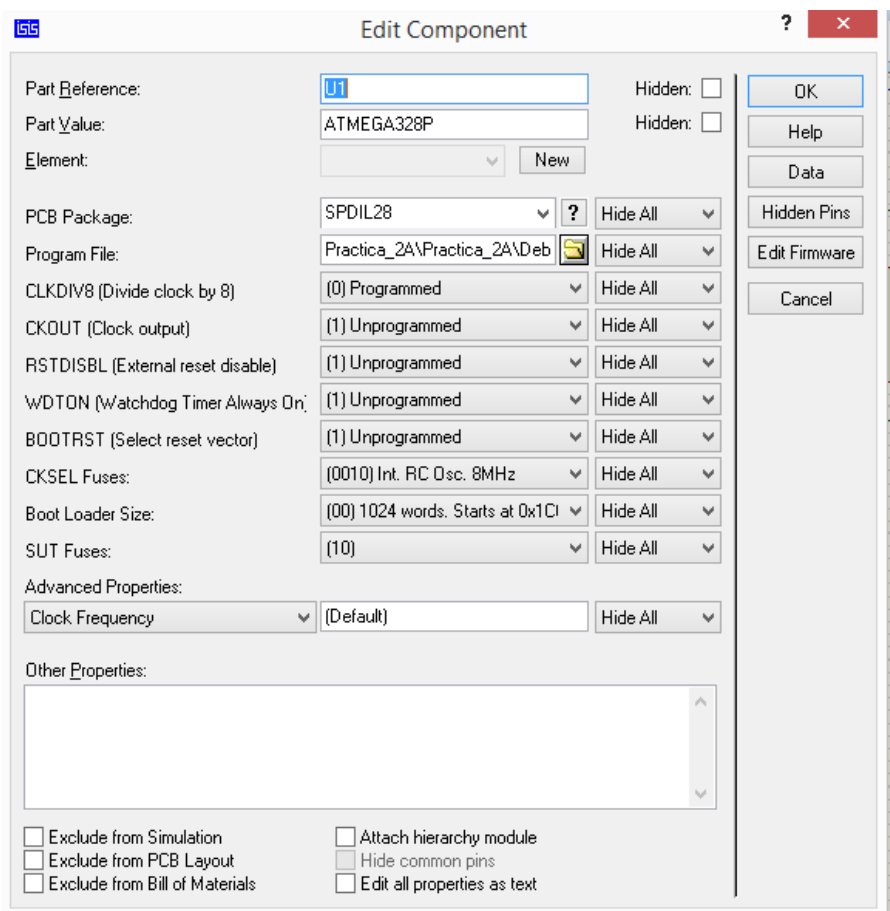
**Figura 3.-** Segmento del código escrito en Atmel Studio.

Una vez escrito el código en Atmel Studio se seleccionó la opción de construir para generar el archivo .hex.



**Figura 4.-** Localización de la opción construir.

Después de generar el archivo .hex, se procedió a cargar este archivo al circuito simulado en proteus para lo cual basta con dar doble clic sobre el microcontrolador simulado y se abrirá una ventana en la cual seleccionaremos el archivo que deseamos montar, en este caso es el correspondiente al ejercicio a de la práctica 2 tal como se muestra en la **figura 5**.



**Figura 5.-** Selección del archivo .hex para simularlo en proteus.

Para realizar la simulación se realizan los mismos pasos para todos los ejercicios por lo que se omite la explicación posterior del procedimiento.

Los resultados obtenidos en la simulación se presentan en la sección de resultados de esta práctica en el apartado que corresponde al primer ejercicio de la práctica.

Después de corroborar que la programación era la correcta se procedió a cargar el archivo .hex al microcontrolador por lo que se utilizó, para este fin, el software extreme Burner y el programador usbasp teniendo cuidado de conectar correctamente cada una de las terminales del microcontrolador.

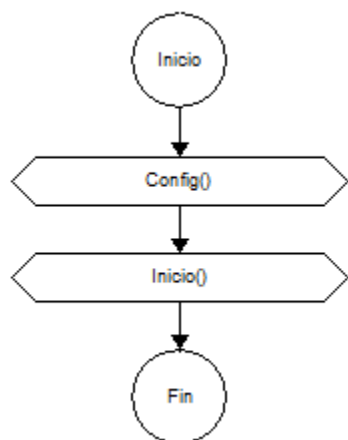
En este paso se colocó el microcontrolador de nuevo en el circuito armado para probar su correcto funcionamiento alimentando el circuito con una fuente de 5 V y obteniendo resultados positivos en el funcionamiento del circuito.

Los resultados obtenidos en este primer ejercicio de la práctica están documentados en la sección de resultados en su apartado correspondiente para el primer ejercicio.

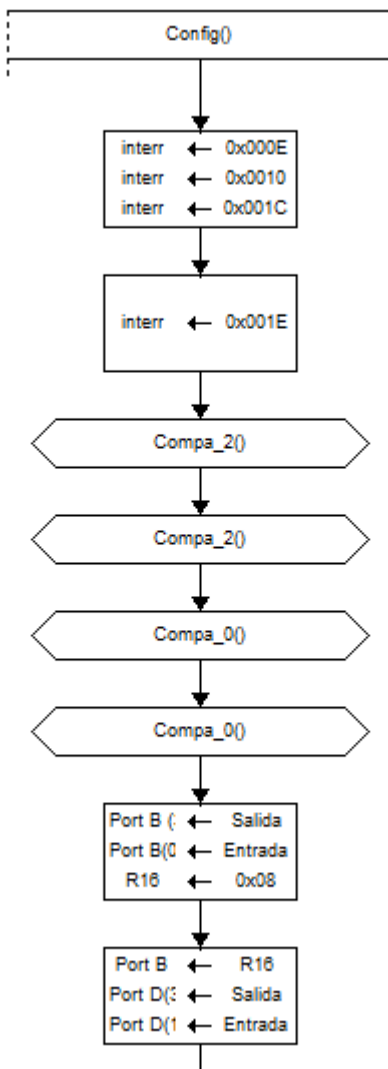
## **5.2.- CONTROL DE UN LED RGB**

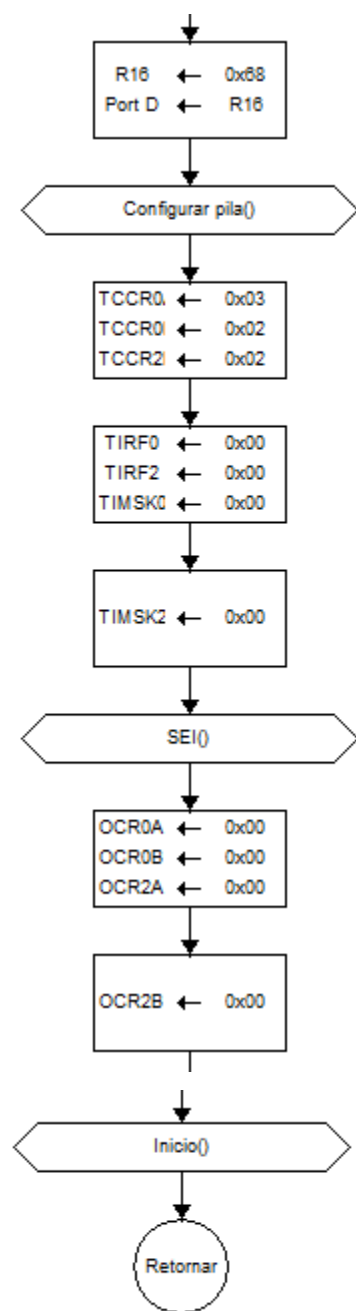
### **5.2.1.- DIAGRAMA DE FLUJO**

A continuación se presenta el diagrama de flujo correspondiente al segundo ejercicio de la práctica 2.



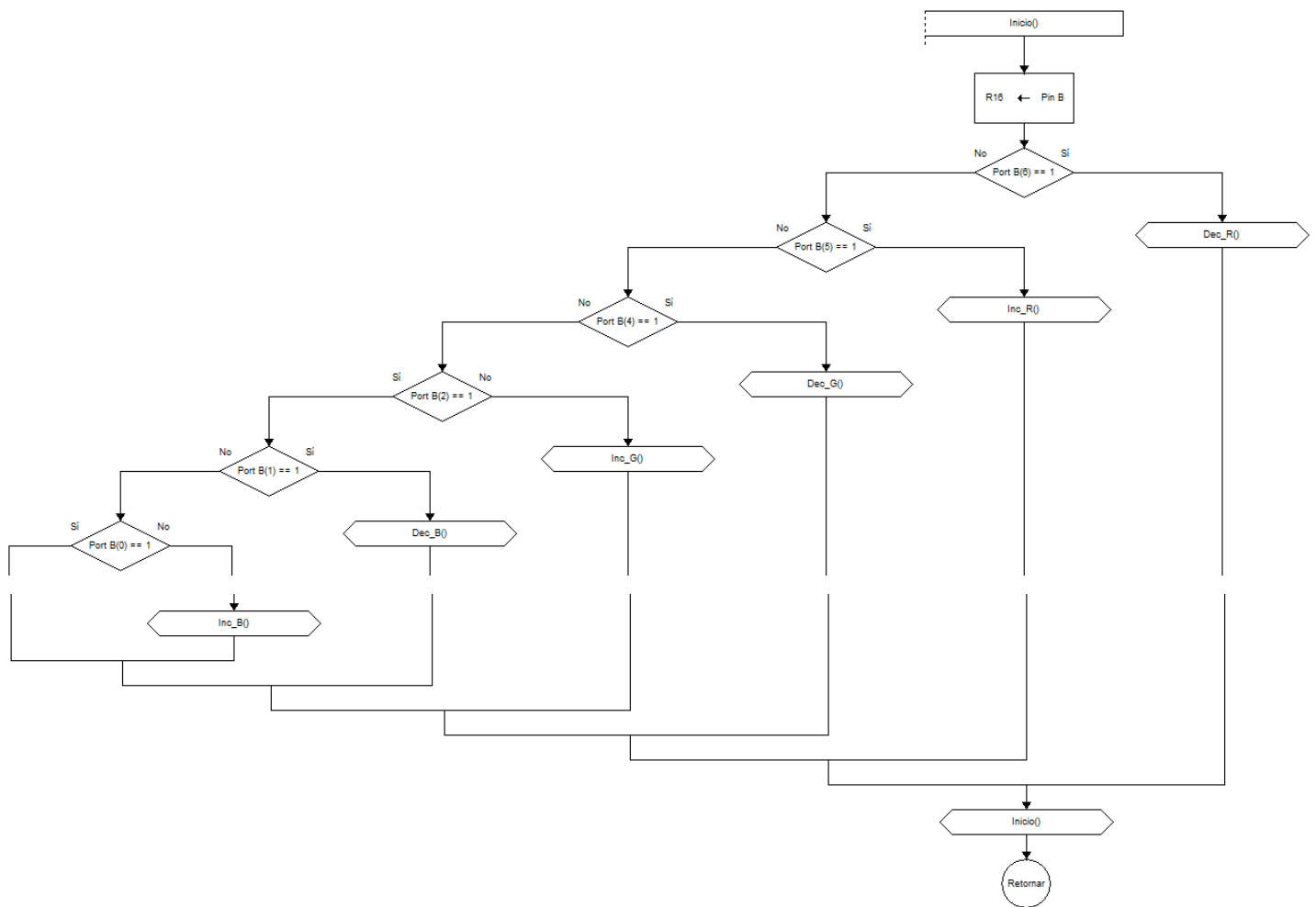
A continuación se presenta la subrutina Config.



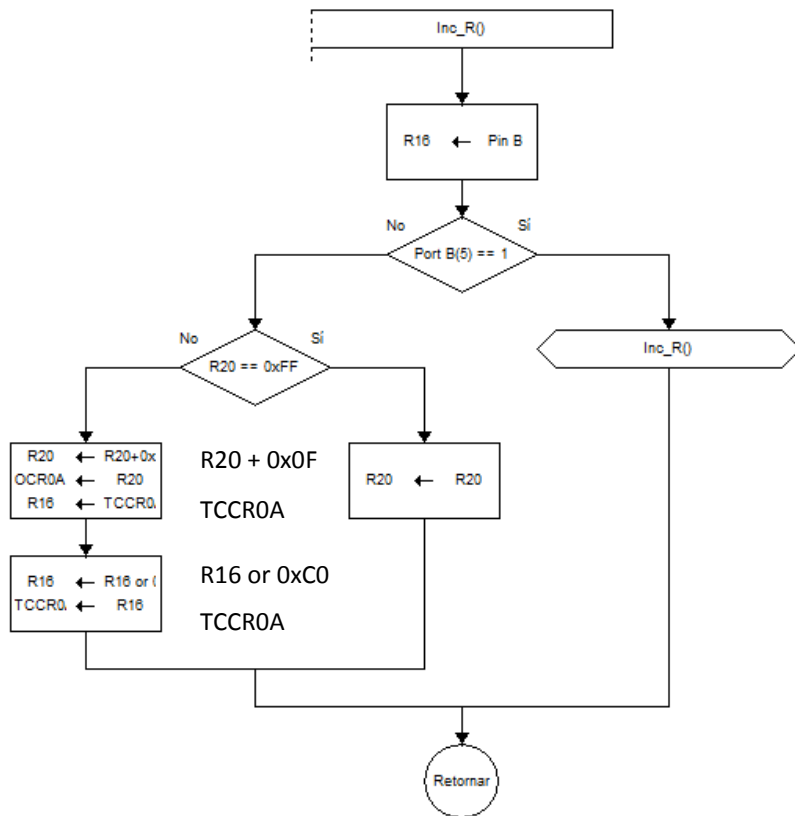


A continuación se presenta la subrutina Inicio.

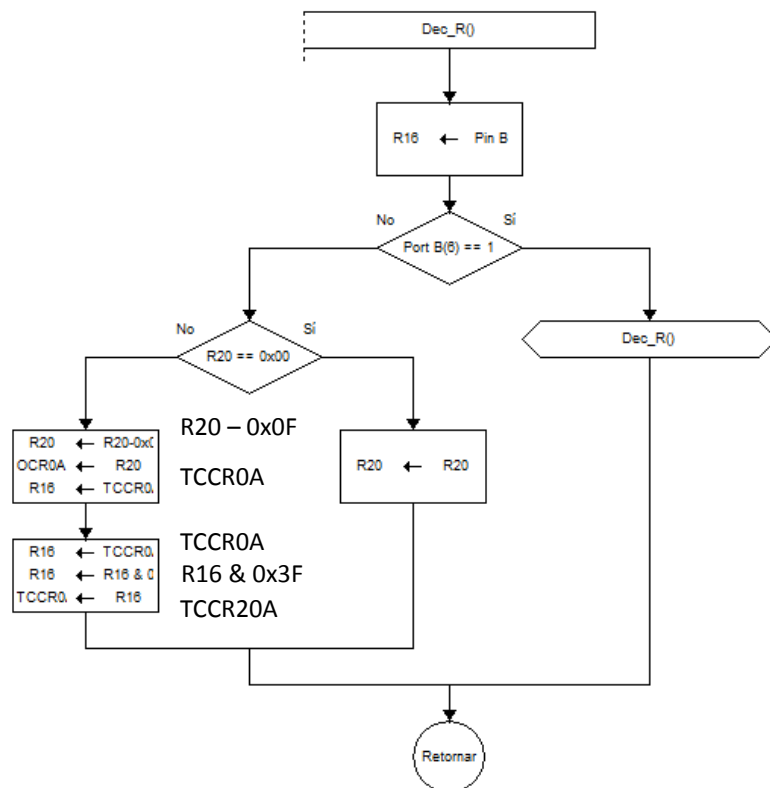




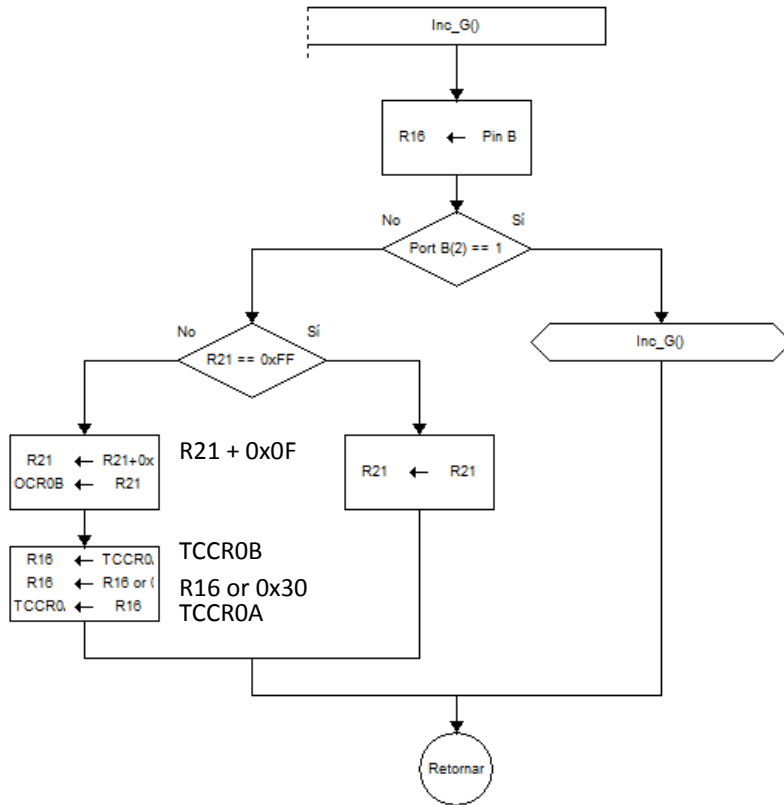
A continuación se presenta la subrutina Inc\_R.



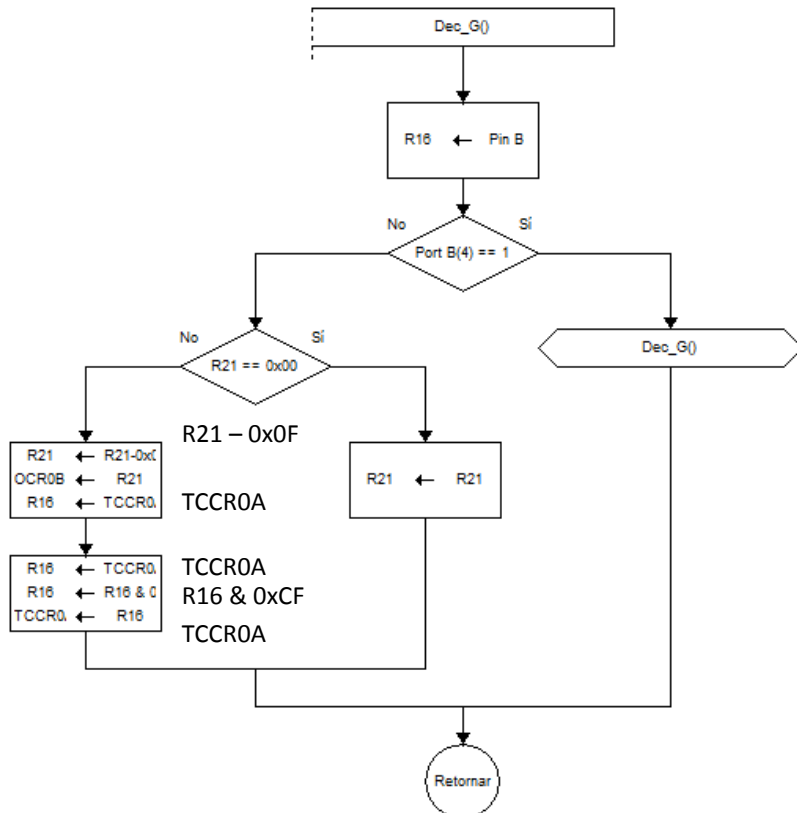
A continuación se presenta la subrutina Dec\_R.



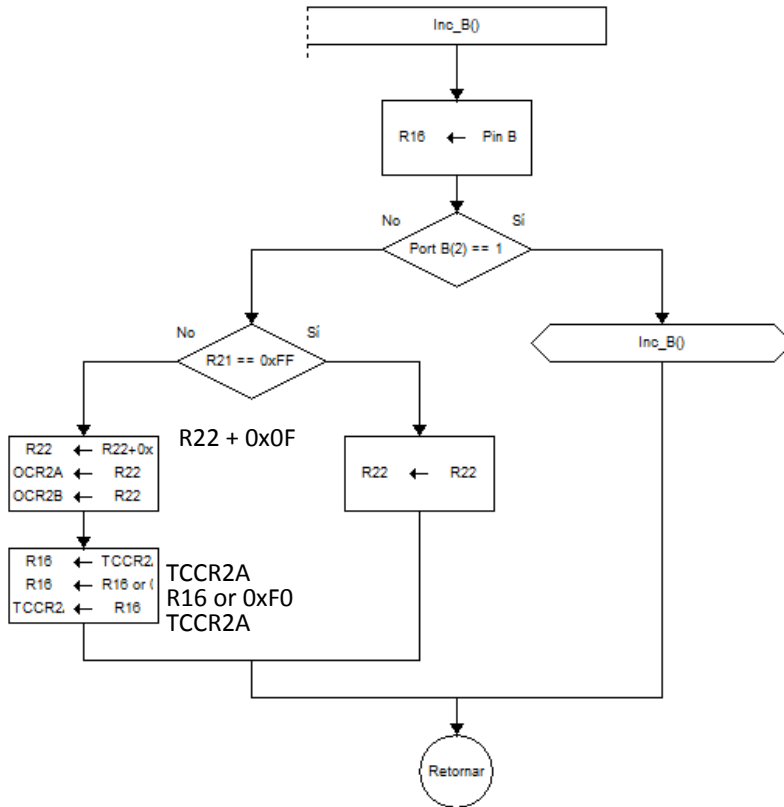
A continuación se presenta la subrutina Inc\_G.



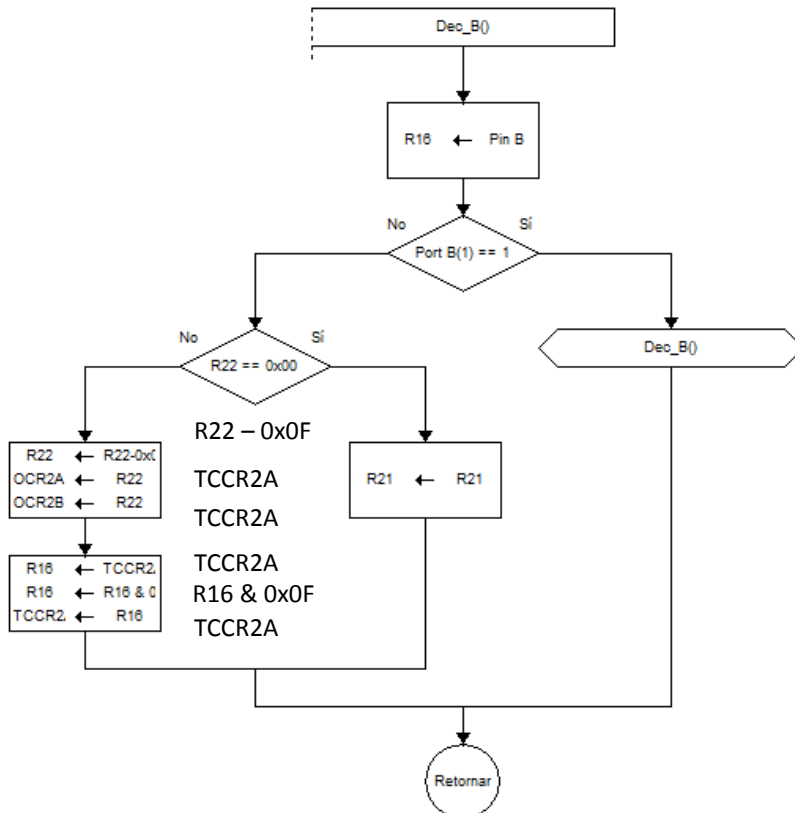
A continuación se presenta la subrutina Dec\_G.



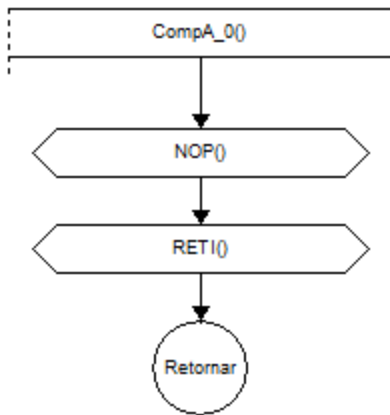
A continuación se presenta la subrutina Inc\_B.



A continuación se presenta la subrutina Dec\_B.



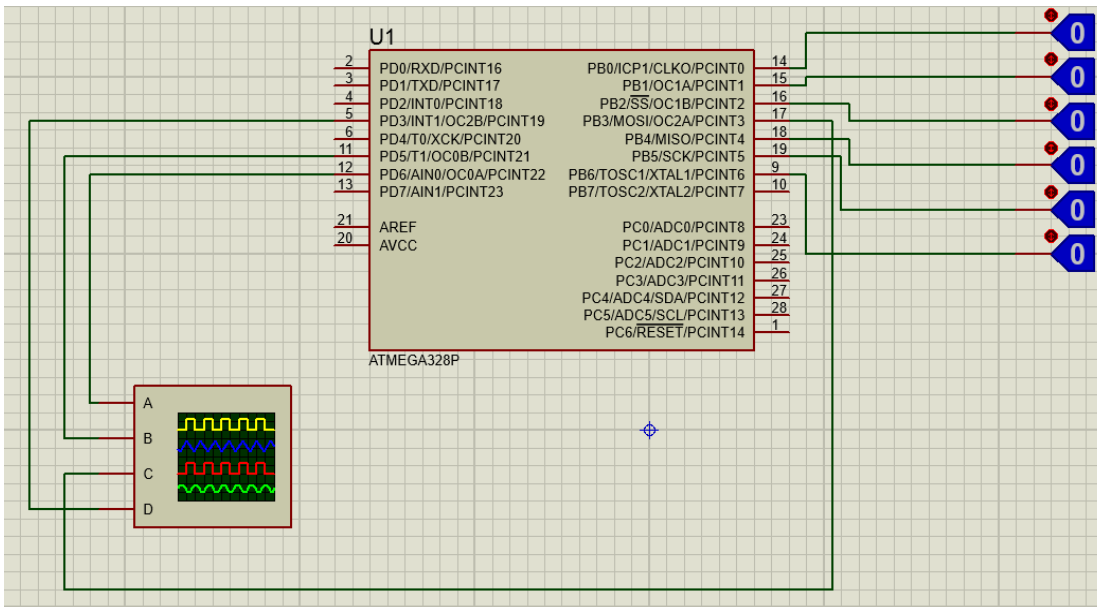
Para el caso de las subrutinas CompA1, B1, A2 y B2, se utiliza el mismo código el cual se ejemplifica solo con la subrutina CompA\_1 que a continuación se muestra.



Este fue el diagrama de flujo completo del segundo ejercicio de esta práctica, y como el software utilizado no permitía una extensión muy larga de texto para ser mostrado en las figuras utilizadas, se crearon cuadros de texto a un lado de las instrucciones que no quedaron visibles para hacer la aclaración del contenido de dichas figuras.

### 5.2.2.- DIAGRAMA ELÉCTRICO

A continuación se presenta el diagrama del circuito a realizar en el laboratorio.



**Figura 6.-** Circuito a realizar en el laboratorio correspondiente al segundo ejercicio.

Como se observa en la **figura 6** se utilizaron en la simulación estados lógicos para simplificar el circuito simulado pero en la práctica se conectaron push buttons.

### 5.2.3.-CÁLCULOS

$$Fp = \frac{1 * 10^6}{8 * (256)} = \frac{15625}{32} = 488.28125Hz$$

$$V_{efectivo} = 5V * \frac{Comparador}{255}$$

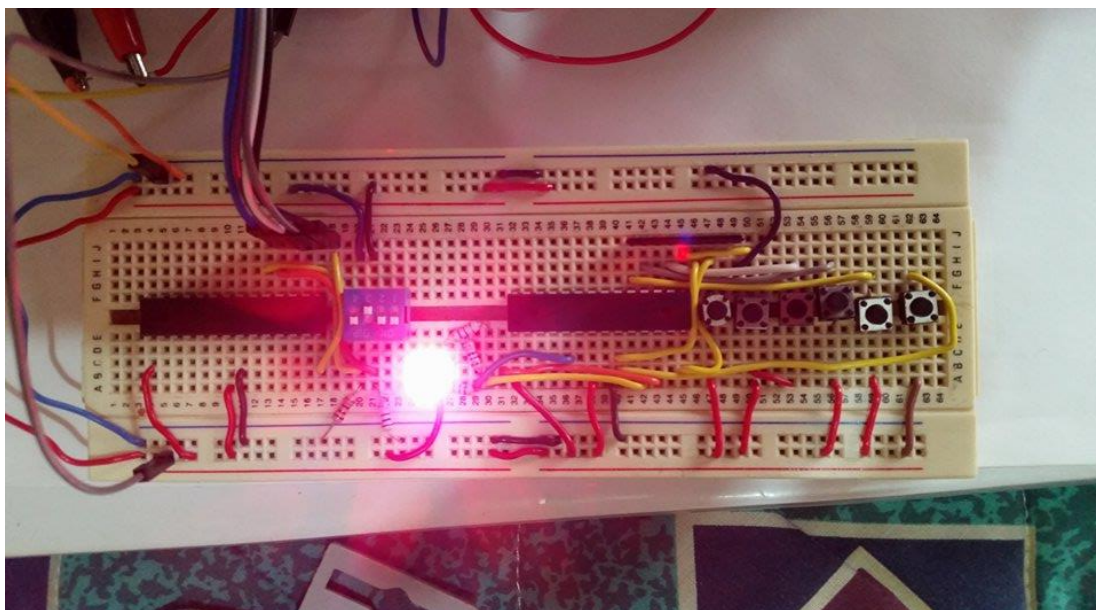
$$Resolución = 5V * \frac{1}{255} \approx 0.294117647 \text{ Volts}$$

Comparador	V <sub>efectivo</sub> [Volts]
00	0
15	0.294117647
30	0.588235294
45	0.882352941
60	1.176470588
75	1.470588235
90	1.764705882
105	2.058823529
120	2.352941176
135	2.647058824
150	2.941176471
165	3.235294118
180	3.529411765
195	3.823529412
210	4.117647059
225	4.411764706
240	4.705882353
255	5

### 5.2.4.- PROCEDIMIENTO

1. Con la ayuda del timer/counter 0, en la configuración fast PWM, realizar el control de intensidad de luz de un led RGB. La intensidad debe de controlarse con dos botones (int- y int+).

Para este segundo ejercicio se armó el circuito que a continuación se presenta.

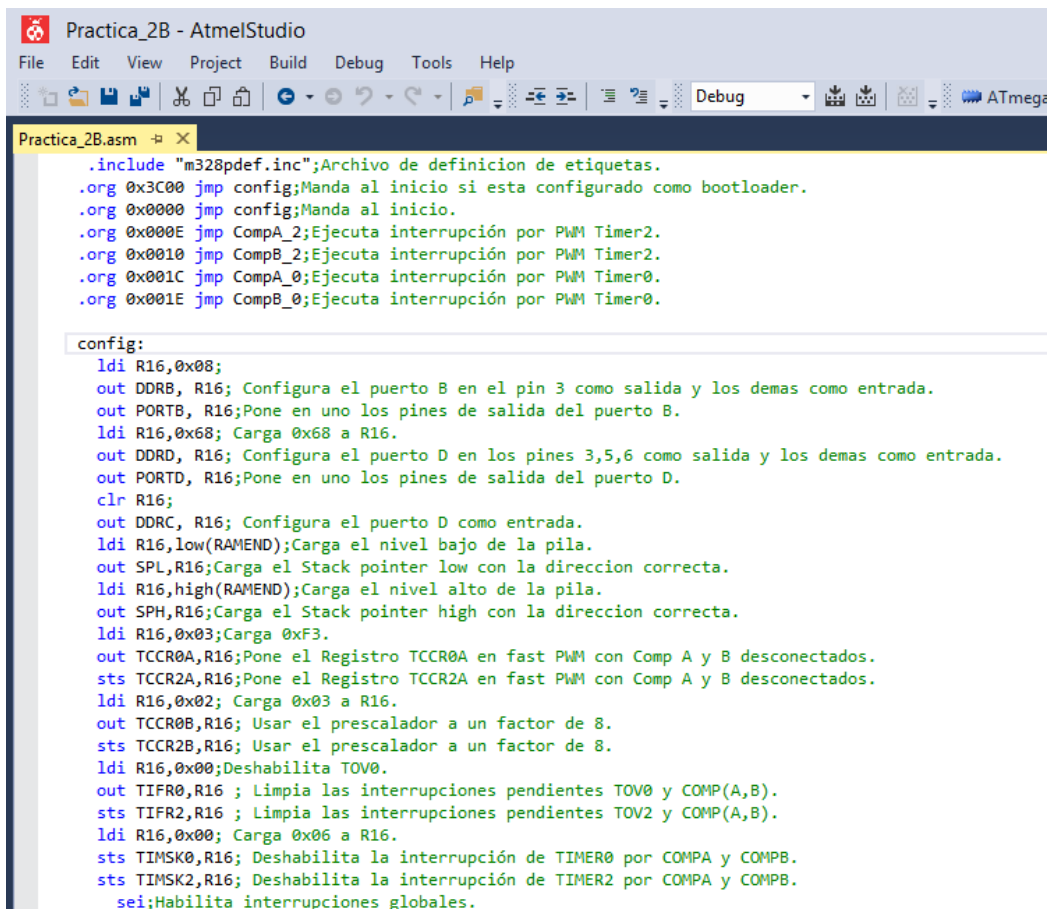


**Figura 7.-** Circuito correspondiente al segundo ejercicio de la práctica 2.

Como se puede observar en la **figura 7** se cuenta con 1 led RGB y con sus respectivos pulsadores para poder aumentar o disminuir la intensidad de cada uno de los colores que lo conforman.

Es necesario aclarar que los procedimientos para generar los archivos y simular los circuitos son exactamente los mismos que en el ejercicio 1 con la salvedad de que el nombre que se le dio a estos nuevos archivos es practica\_2B por lo que se omitirá la explicación de todos los pasos que pueden ser consultados en el procedimiento del ejercicio 1.

A continuación se muestra un segmento del código que se escribió en el software Atmel Studio para este segundo ejercicio simplemente para documentar este paso puesto que el código difiere respecto del código del ejercicio 1.



```
.include "m328pdef.inc"; Archivo de definicion de etiquetas.
.org 0x3C00 jmp config;Manda al inicio si esta configurado como bootloader.
.org 0x0000 jmp config;Manda al inicio.
.org 0x000E jmp CompA_2;Ejecuta interrupción por PWM Timer2.
.org 0x0010 jmp CompB_2;Ejecuta interrupción por PWM Timer2.
.org 0x001C jmp CompA_0;Ejecuta interrupción por PWM Timer0.
.org 0x001E jmp CompB_0;Ejecuta interrupción por PWM Timer0.

config:
    ldi R16,0x08;
    out DDRB, R16; Configura el puerto B en el pin 3 como salida y los demas como entrada.
    out PORTB, R16;Pone en uno los pines de salida del puerto B.
    ldi R16,0x68; Carga 0x68 a R16.
    out DDRD, R16; Configura el puerto D en los pines 3,5,6 como salida y los demas como entrada.
    out PORTD, R16;Pone en uno los pines de salida del puerto D.
    clr R16;
    out DDRC, R16; Configura el puerto D como entrada.
    ldi R16,low(RAMEND);Carga el nivel bajo de la pila.
    out SPL,R16;Carga el Stack pointer low con la direccion correcta.
    ldi R16,high(RAMEND);Carga el nivel alto de la pila.
    out SPH,R16;Carga el Stack pointer high con la direccion correcta.
    ldi R16,0x03;Carga 0xF3.
    out TCCR0A,R16;Pone el Registro TCCR0A en fast PWM con Comp A y B desconectados.
    sts TCCR2A,R16;Pone el Registro TCCR2A en fast PWM con Comp A y B desconectados.
    ldi R16,0x02; Carga 0x03 a R16.
    out TCCR0B,R16; Usar el prescalador a un factor de 8.
    sts TCCR2B,R16; Usar el prescalador a un factor de 8.
    ldi R16,0x00;Deshabilita TOV0.
    out TIFR0,R16 ; Limpia las interrupciones pendientes TOV0 y COMP(A,B).
    sts TIFR2,R16 ; Limpia las interrupciones pendientes TOV2 y COMP(A,B).
    ldi R16,0x00; Carga 0x06 a R16.
    sts TIMSK0,R16; Deshabilita la interrupción de TIMER0 por COMPA y COMPB.
    sts TIMSK2,R16; Deshabilita la interrupción de TIMER2 por COMPA y COMPB.
    sei;Habilita interrupciones globales.
```

**Figura 8.-** Segmento de código del segundo ejercicio.

El archivo que contiene el código fue agregado junto con este reporte por lo que se omitió colocar más código en la práctica.

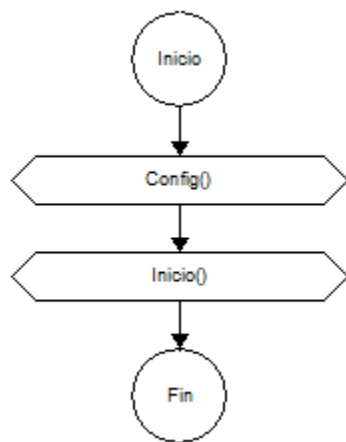
Los resultados obtenidos en este ejercicio se encuentran en su respectivo apartado en la sección de resultados.

## 5.3.- GENERADOR DE FRECUENCIAS

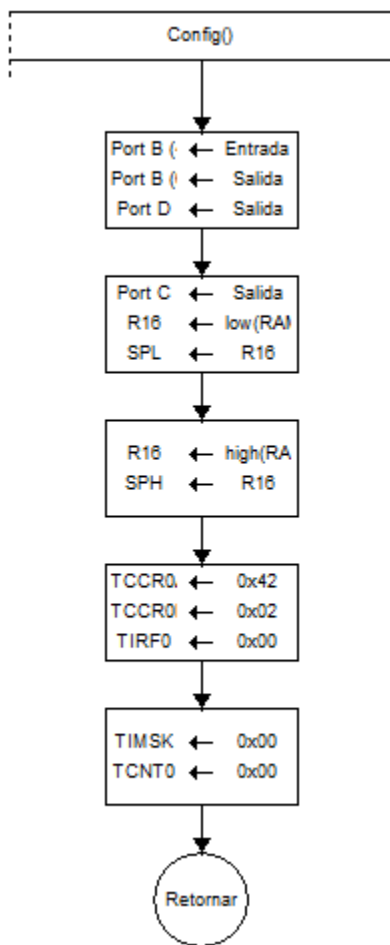
### 5.3.1.- DIAGRAMA DE FLUJO

A continuación se presenta el diagrama de flujo correspondiente al tercer ejercicio de la práctica 2.

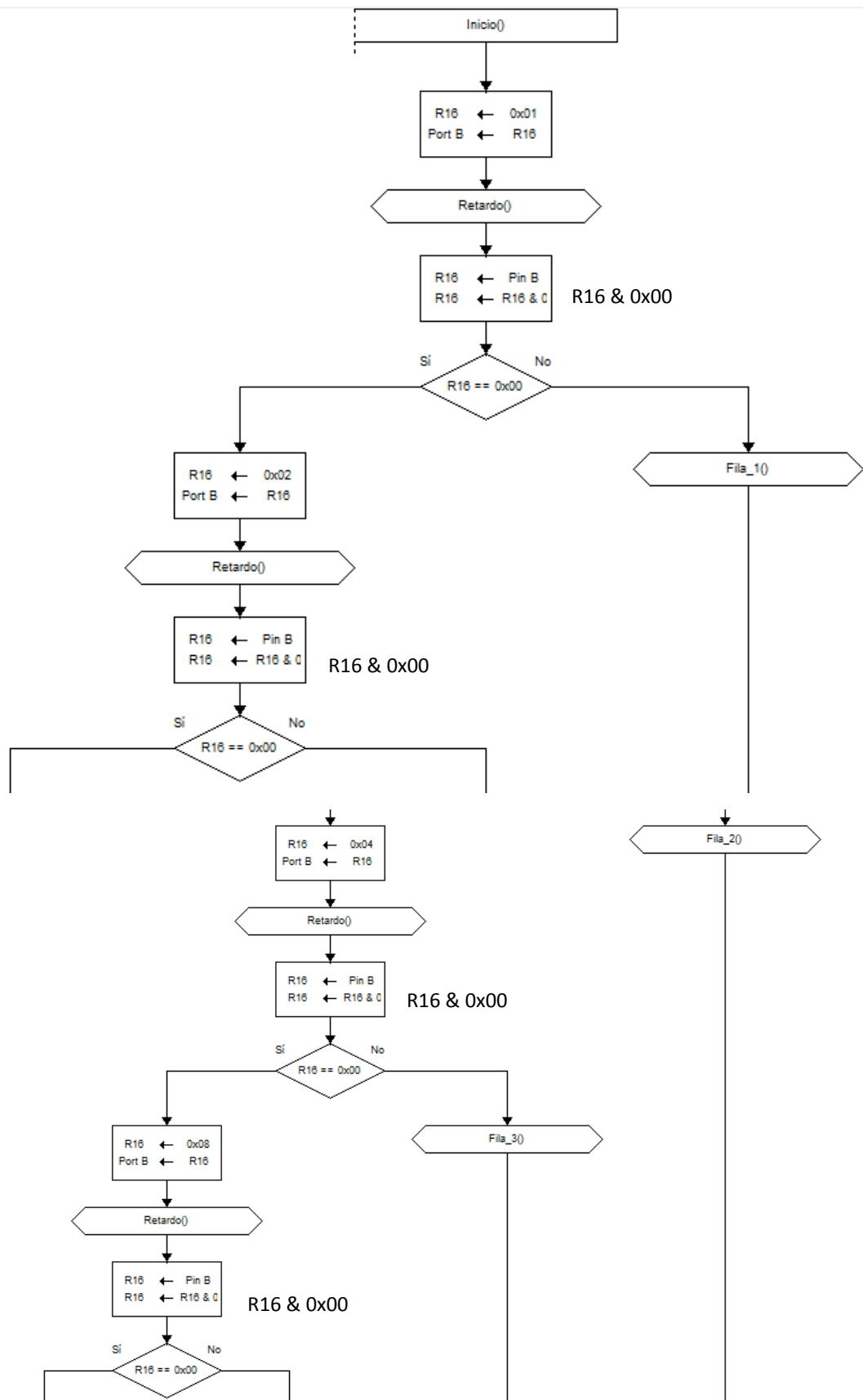


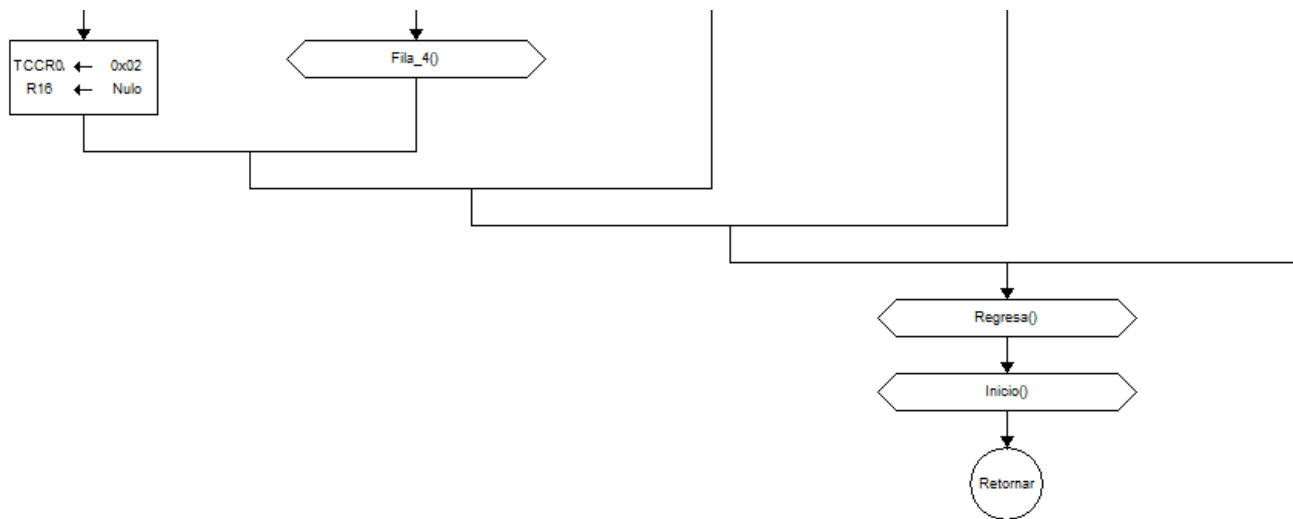


A continuación se presenta la subrutina config.



A continuación se presenta la subrutina de inicio que es en la cual se envían y reciben los datos del teclado matricial.



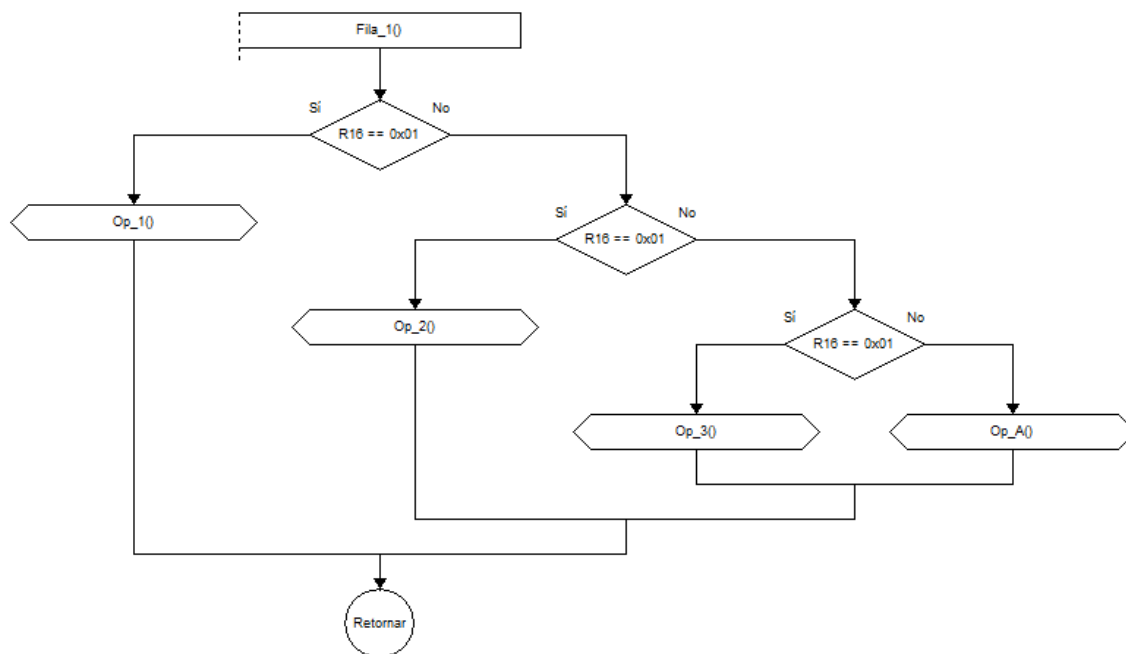


En la rutina anterior se encuentran algunos valores que no fue posible visualizar dentro de los cuadros de operación sin embargo se agregaron cuadros de texto para hacer la aclaración correspondiente del valor que debe de tener.

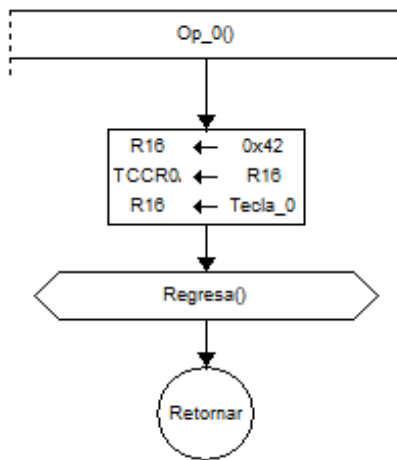
Todas y cada una de las subrutinas de fiala realizan la misma operación con la salvedad de que los datos que se reciben se interpretan de diferente forma de acuerdo a la tecla presionada por lo que solo se explicará una subrutina para ejemplificar todas.

A continuación se presenta la subrutina correspondiente a la fila 1.

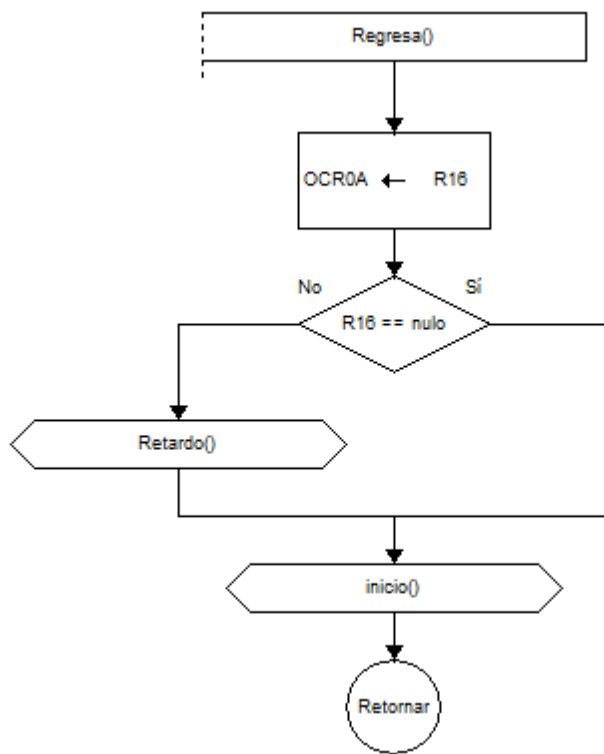
Como se observa en esta subrutina, dependiendo de que tecla se presione nos enviará a alguna subrutina en las cuales se define que valor tendrá nuestro registro de comparación para poder lograr la frecuencia deseada.



A continuación se presenta la subrutina Op\_0.



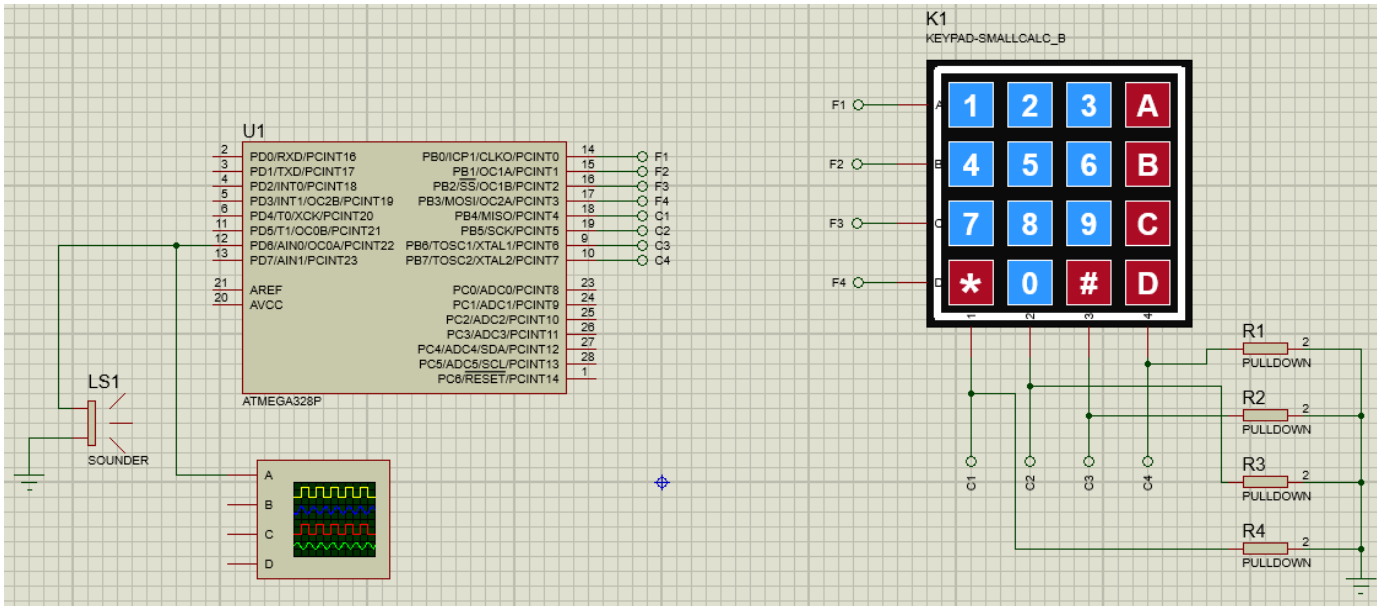
A continuación se presenta la subrutina Regresa.



Este fue el diagrama de flujo del tercer ejercicio de esta práctica.

### 5.3.2.- DIAGRAMA ELÉCTRICO

A continuación se presenta el diagrama del circuito a realizar en el laboratorio.



**Figura 9.-** Circuito a realizar en el laboratorio correspondiente al tercer ejercicio.

### 5.2.3.-CÁLCULOS

$$F_{max} = \frac{1 * 10^6}{2 * 8 * (1)} = 62500 \text{ Hz}$$

$$F_{min} = \frac{1 * 10^6}{2 * 8 * (256)} \approx 244.1406 \text{ Hz}$$

$$F = \frac{1 * 10^6}{2 * 8 * (1 + \text{Comparador})}$$

$$\text{Comparador} = \frac{1 * 10^6}{F * 2 * 8} - 1$$


Tecla	Frecuencia	Comparador	Carga Real	Frecuencia Real [Hz]
0	493.88	125.548959	126	492.125984
1	466.16	133.074138	133	466.41791
2	440.40	140.91644	141	440.140845
3	415.30	149.493619	149	416.666667
4	392.00	158.438776	158	393.081761
5	369.99	167.923484	168	369.822485
6	349.23	177.965152	178	349.162011
7	329.63	188.606529	189	328.947368

Tecla	Frecuencia	Comparador	Carga Real	Frecuencia Real [Hz]
8	311.13	199.880661	200	310.945274
9	293.66	211.831165	212	293.42723
A	277.18	224.485244	224	277.777778
B	261.63	237.886978	238	261.506276

### 5.3.4.- PROCEDIMIENTO

1. Realizar un generador de frecuencias con la ayuda de la interrupción timer CTC. Este generador debe de tener las frecuencias de las notas musicales como se muestra en la figura, de tal manera que se elabore un pequeño teclado. Este teclado cada vez que se presione un botón, debe de sonar la nota musical 0.5 segundos, mostrar las frecuencias en un osciloscopio.

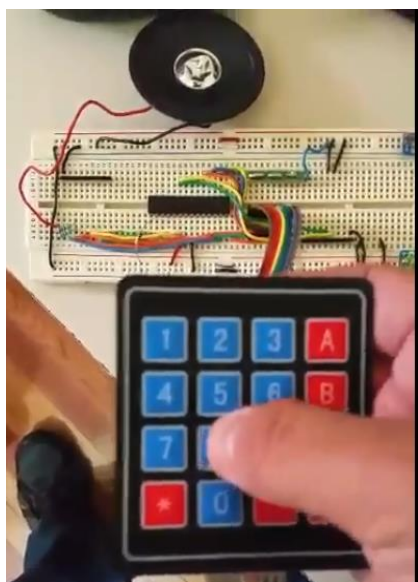
En la siguiente imagen se puede observar las frecuencias a generar.



B	Si	493.88
A# (Bb)	La # (Si b)	466.16
A	La	440.00
G# (Ab)	Sol # (La b)	415.30
G	Sol	392.00
F# (Gb)	Fa # (Sol b)	369.99
F	Fa	349.23
E	Mi	329.63
D# (Eb)	Re # (Mi b)	311.13
D	Re	293.66
C# (Db)	Do # (Re b)	277.18
C	Do	261.63

**Figura 10.-** Frecuencias a reproducir.

Para este tercer ejercicio se armó el circuito que a continuación se presenta.



**Figura 11.-** Circuito correspondiente al tercer ejercicio de la práctica 2.

A continuación se muestra un segmento del código que se escribió en el software Atmel Studio para este tercer ejercicio.

```
Practica_2C - AtmelStudio
File Edit View Project Build Debug Tools Help
Practica_2C.asm*
.include "m328pdef.inc"; Archivo de definicion de etiquetas.
.org 0x3C00 jmp config; Manda al inicio si esta configurado como bootloader.
.org 0x0000 jmp config; Manda al inicio
.equ Tecla_N=0x00; Define el valor de la Tecla Nula.
.equ Tecla_0=0x7E; Define el valor de la constante para generar un "0" .
.equ Tecla_1=0x85; Define el valor de la constante para generar un "1" .
.equ Tecla_2=0x8D; Define el valor de la constante para generar un "2" .
.equ Tecla_3=0x95; Define el valor de la constante para generar un "3" .
.equ Tecla_4=0x9E; Define el valor de la constante para generar un "4" .
.equ Tecla_5=0xA8; Define el valor de la constante para generar un "5" .
.equ Tecla_6=0xB2; Define el valor de la constante para generar un "6" .
.equ Tecla_7=0xBD; Define el valor de la constante para generar un "7" .
.equ Tecla_8=0xD4; Define el valor de la constante para generar un "8" .
.equ Tecla_9=0xE0; Define el valor de la constante para generar un "9" .
.equ Tecla_A=0xEE; Define el valor de la constante para generar una "A" .
.equ Tecla_B=0x00; Define el valor de la constante para generar una "B" .
.equ Tecla_C=0x00; Define el valor de la constante para generar una "C" .
.equ Tecla_D=0x00; Define el valor de la constante para generar una "D" .
.equ Tecla_E=0x00; Define el valor de la constante para generar una "E" .
.equ Tecla_F=0x00; Define el valor de la constante para generar una "F" .
```

**Figura 12.-** Segmento de código del tercer ejercicio.

El archivo que contiene el código fue agregado junto con este reporte por lo que se omitió colocar más código en la práctica.

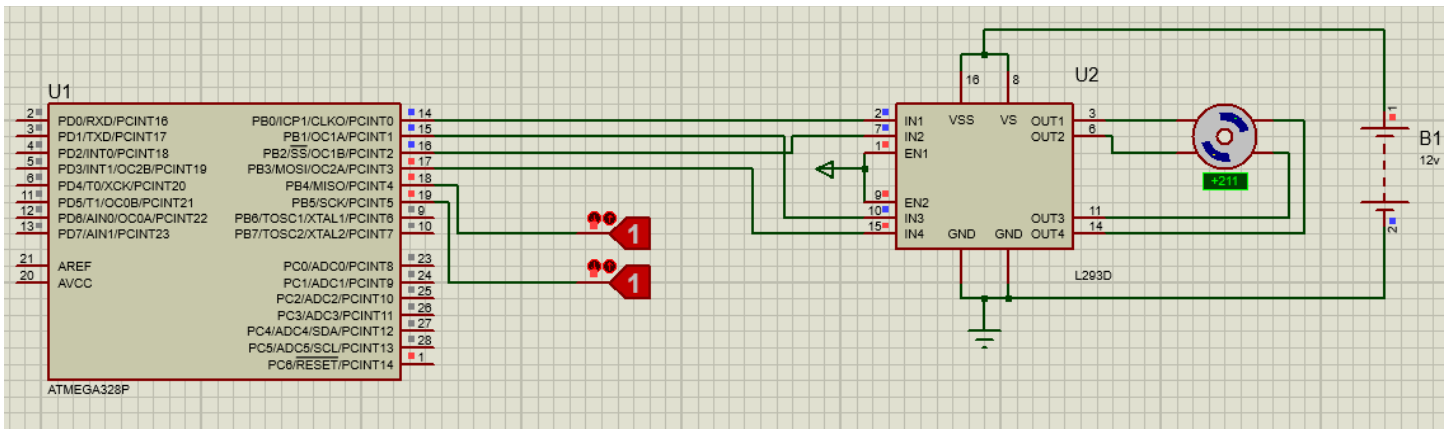
Tanto para éste ejercicio como para el segundo, se siguieron los mismos pasos para poder crear el archivo de simulación y para poder cargar los archivos necesarios para correr la simulación por lo que no se explican de nuevo estos procedimientos.

Los resultados obtenidos en este ejercicio se encuentran en su respectivo apartado en la sección de resultados.

## 6.- RESULTADOS

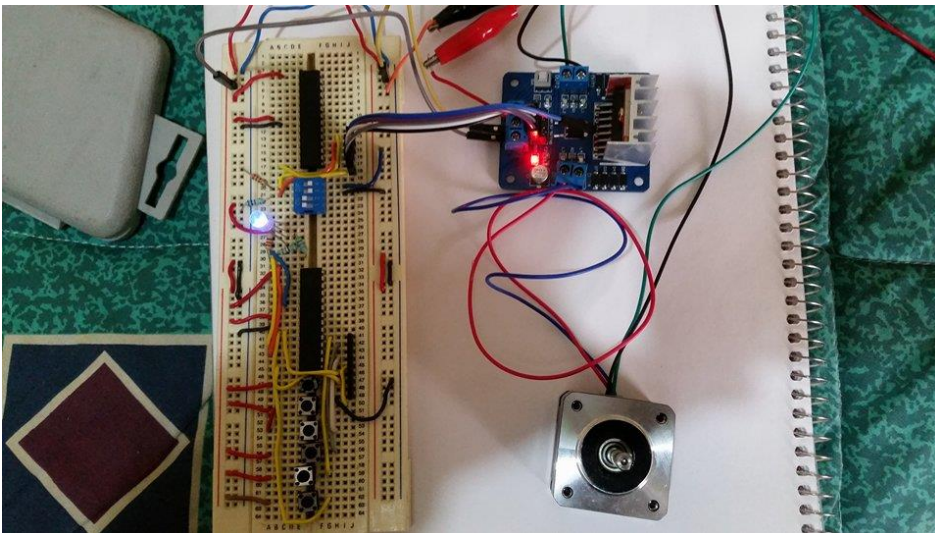
### 6.1.- CONTROL DE UN MOTOR A PASOS

Para comprobar que la programación era la correcta, se simuló el comportamiento que tendría el microcontrolador en proteus y tal como se esperaba, la programación fue correcta, no es posible visualizar el giro del motor pero se coloca una imagen de todos modos.



**Figura 13.-** Simulación en proteus para el primer ejercicio.

Ahora se observa a continuación una imagen del circuito funcionando en la práctica aunque no se observe, el motor si funciona como se esperaba.



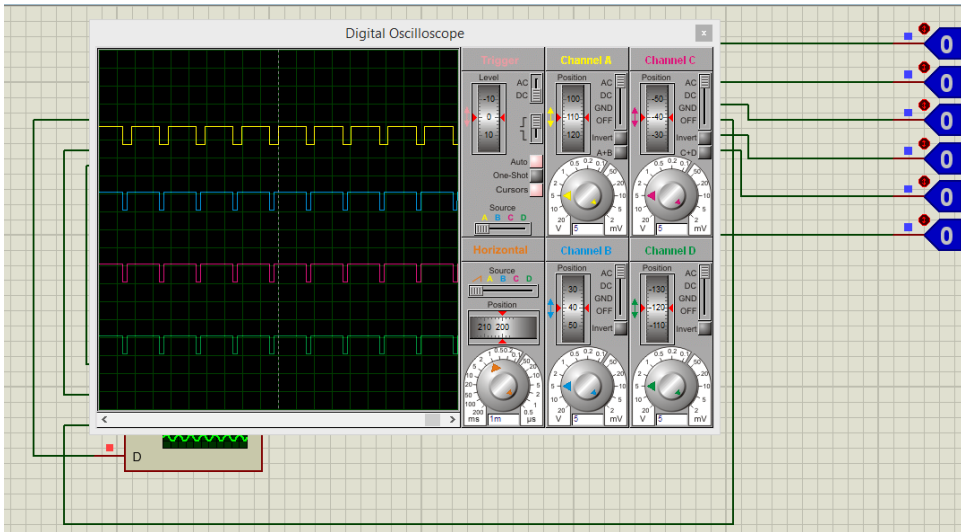
**Figura 14.-** Prueba del ejercicio 1.

Con las figuras anteriores se comprueba el correcto funcionamiento y programación del ejercicio 1 de esta práctica.

## 6.2.- CONTROL DE UN LED RGB

Para comprobar que la programación era la correcta, se simuló el comportamiento que tendría el microcontrolador en proteus y tal como se ve a continuación la programación fue correcta.

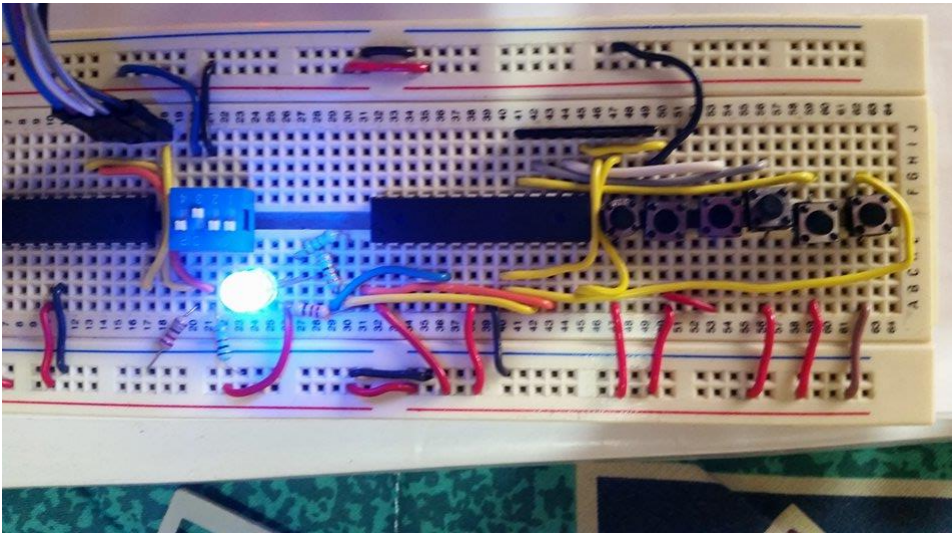




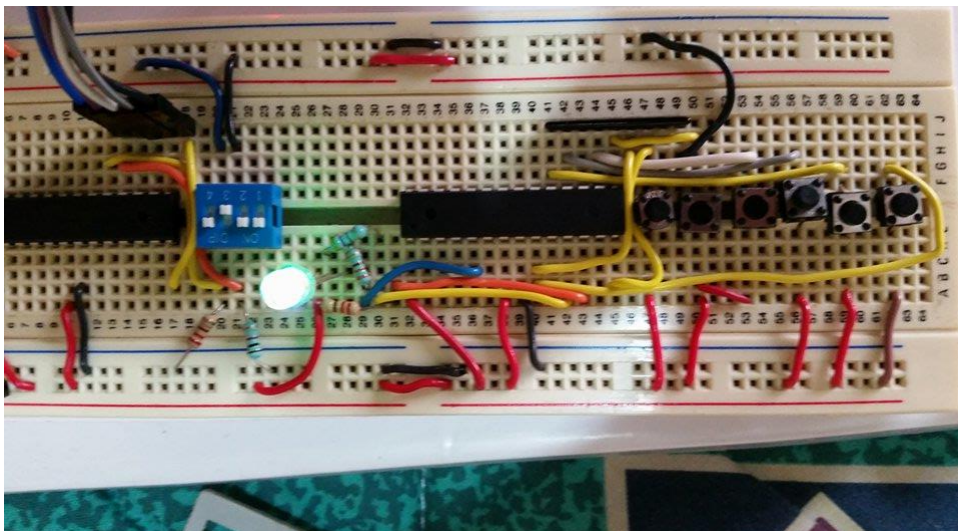
**Figura 15.-** Simulación en proteus para el segundo ejercicio.

Tal como se observa en la **figura 15** es posible variar el ciclo útil del PWM para así generar diferente voltaje a la salida y poder realizar el control de la luminosidad del led.

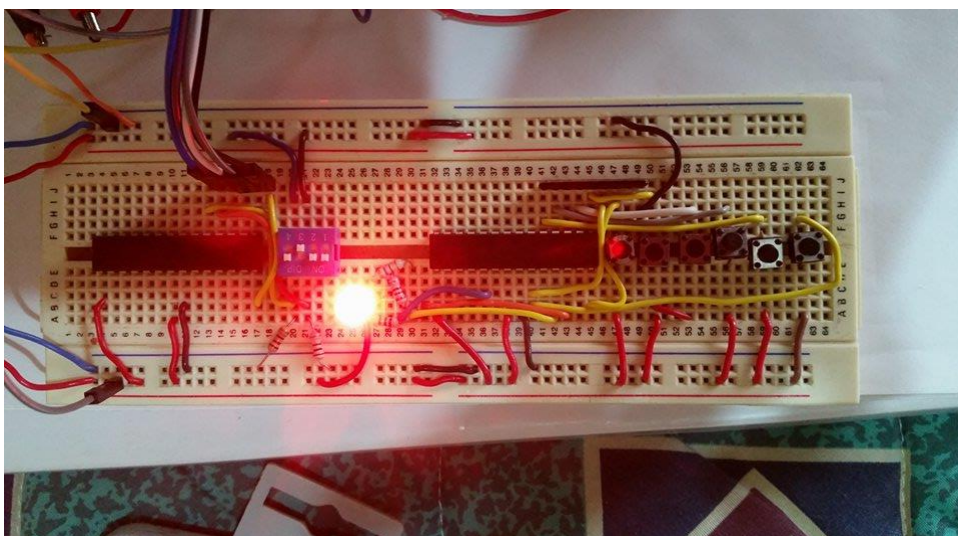
Se puede comprobar un correcto funcionamiento del circuito con las **figuras 16 y 17** en las cuales se observa la distinta coloración y luminosidad del led.



**Figura 16.-** Prueba 1 del ejercicio 2, color azul.



**Figura 17.-** Prueba 2 del ejercicio 2, color verde.

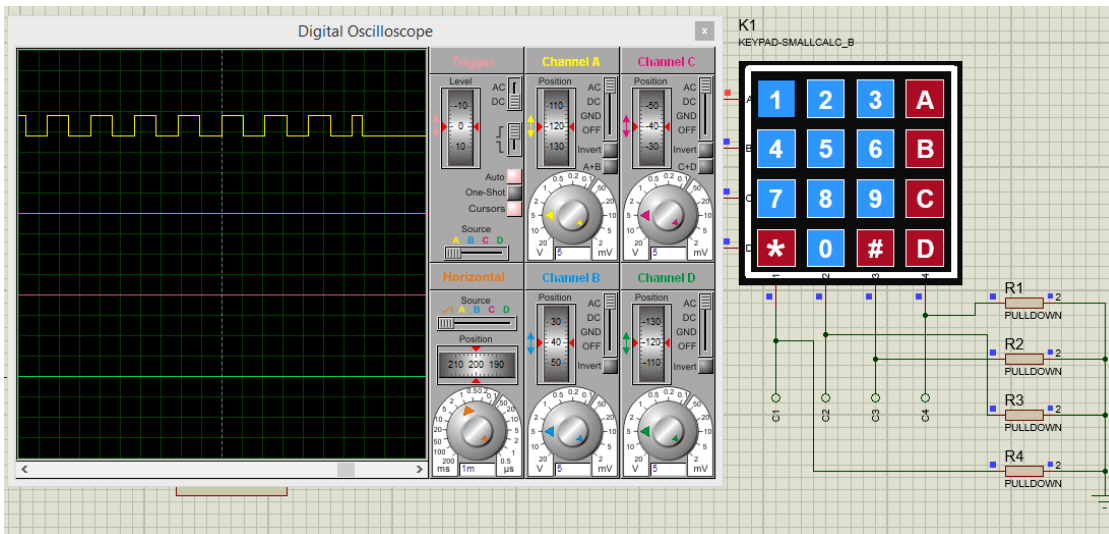


**Figura 18.-** Prueba 3 del ejercicio 2, color rojo.

Con las figuras anteriores se comprueba el correcto funcionamiento y programación del ejercicio 2 de esta práctica.

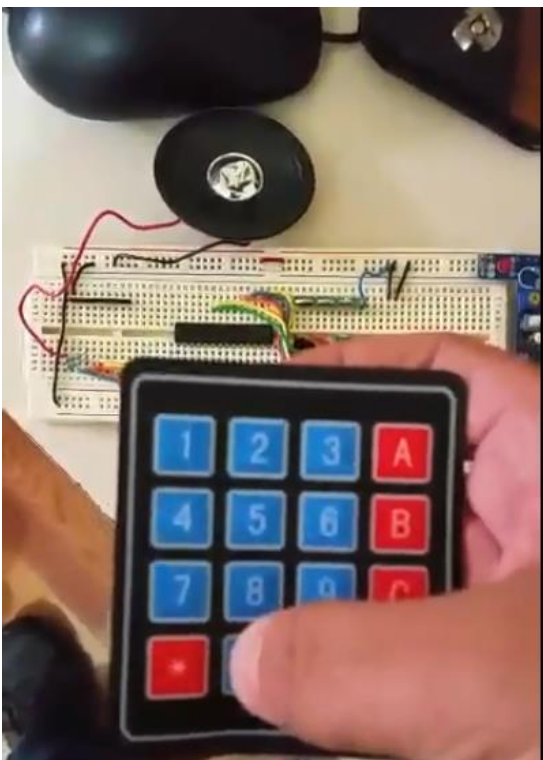
### **6.3.- GENERADOR DE FRECUENCIAS**

Para comprobar que la programación era la correcta, se simuló el comportamiento que tendría el microcontrolador en proteus y tal como se ve a continuación la programación fue correcta.



**Figura 19.-** Simulación en proteus para el tercer ejercicio.

Para éste ejercicio no será posible visualizar el correcto funcionamiento puesto que se trata de generación de sonido.



**Figura 20.-** Prueba del ejercicio 3.

Este ejercicio fue resuelto de acuerdo a las especificaciones aunque en este reporte es muy difícil comprobar su correcto funcionamiento.

## 7.- CONCLUSIONES

### - Aguilar Zarazúa Luis Fernando

Se aplicaron los conceptos teóricos vistos en clase referentes a las interrupciones en particular a la interrupción timer overflow la cual nos permite generar un pulso PWM con el cual es posible realizar varias funciones puesto que podemos variar el valor del voltaje final así como también la frecuencia del pulso.

En el desarrollo de la práctica se tuvo que hacer uso de la señal PWM como una forma de poder controlar algunas propiedades de algunos elementos como lo es la velocidad de un motor a pasos y la luminosidad así como el color de un led rgb.

Se reforzó el conocimiento que tenía al realizar la práctica y dar solución a los problemas propuestos.

### - Martínez Pérez Nestor

En esta práctica se analizaron conceptos nuevos que nos ayudan a la programación de otras funciones prácticas del microcontrolador como lo son las interrupciones. En este caso se trabajó con la interrupción timer con la cual fue posible el desarrollo de los ejercicios.

Se pudo realizar el correcto manejo del modo normal del timer para poder controlar un motor a pasos y se conoció una de las configuraciones que nos ofrecen los registros que se utilizan para configurar esta interrupción.

Se pudo hacer uso de la configuración fast PWM para poder generar diferentes salidas de voltaje producida por el microcontrolador.

Fue posible realizar una variación de frecuencia de un PWM gracias a la comparación entre un registro el valor de nuestro timer.

Las interrupciones ocurren de manera paralela a la ejecución de nuestros programas por lo que en uno de los ejercicios fue posible dejar que una subrutina vacía se ejecutara a si misma dado que todo el procedimiento se realizaba gracias a la interrupción timer y hasta que no ocurría, no se realizaba nada más salvo la lectura de las entradas y la actualización de las salidas.

## 8.- REFERENCIAS DE INTERNET

[1].- <http://polloleeds.blogspot.mx/>