



INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y
TECNOLOGÍAS AVANZADAS

“Práctica 1: Rutinas en AVR”

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.- FRECUENCIA DER RELOJ	1
3.2.- STACK O PILA	2
3.2.1.- PUNTERO A LA PILA (STACK POINTER)	2
3.2.2.- PUSH	2
3.2.3.- POP	2
4.- MATERIAL	3
5.- DESARROLLO EXPERIMENTAL	3
5.1.- SECUENCIAS BINARIAS	3
5.1.1.- DIAGRAMA DE FLUJO	3
5.1.2.- DIAGRAMA ELÉCTRICO	7
5.1.3.- PROCEDIMIENTO.....	7
5.2.- SEGUNDERO (0 - 60)	10
5.2.1.- DIAGRAMA DE FLUJO	10
5.2.2.- DIAGRAMA ELÉCTRICO	14
5.2.3.- PROCEDIMIENTO.....	14
5.3.- TECLADO MATRICIAL 4x4	16
5.3.1.- DIAGRAMA DE FLUJO	16
5.3.2.- DIAGRAMA ELÉCTRICO	19
5.3.3.- PROCEDIMIENTO.....	20
6.- RESULTADOS	21
6.1.- SECUENCIAS BINARIAS	21
6.2.- SEGUNDERO (0 - 60)	23
6.3.- TECLADO MATRICIAL 4x4	25
7.- CONCLUSIONES.....	28
- Aguilar Zarazúa Luis Fernando	28
- Martínez Pérez Nestor	28
8.- REFERENCIAS DE INTERNET	28

ÍNDICE DE FIGURAS

Figura 1.- Circuito a realizar en el laboratorio.	7
Figura 2.- Circuito correspondiente al primer ejercicio de la práctica 1.	8
Figura 3.- Segmento del código escrito en Atmel Studio.	8
Figura 4.- Localización de la opción construir.	9
Figura 5.- Selección del archivo .hex para simularlo en proteus.	9
Figura 6.- Circuito a realizar en el laboratorio correspondiente al segundo ejercicio.	14
Figura 7.- Configuración de un display de 7 segmentos.	15
Figura 8.- Circuito correspondiente al segundo ejercicio de la práctica 1.	15
Figura 9.- Segmento de código del segundo ejercicio.	16
Figura 10.- Circuito a realizar en el laboratorio correspondiente al tercer ejercicio.	19
Figura 11.- Configuración de un teclado matricial 4x4.	20
Figura 12.- Circuito correspondiente al tercer ejercicio de la práctica 1.	20
Figura 13.- Segmento de código del tercer ejercicio.	21
Figura 14.- Simulación en proteus para el primer ejercicio, secuencia 2.	22
Figura 15.- Simulación en proteus para el primer ejercicio, secuencia 3.	22
Figura 16.- Prueba 1 del ejercicio 1, secuencia 2.	23
Figura 17.- Prueba 2 del ejercicio 1, secuencia 3.	23
Figura 18.- Simulación en proteus para el segundo ejercicio, reset activado.	24
Figura 19.- Simulación en proteus para el segundo ejercicio, pausa activada.	24
Figura 20.- Prueba 1 del ejercicio 2, reset activado.	25
Figura 21.- Prueba 2 del ejercicio 2, pausa activada.	25
Figura 22.- Simulación en proteus para el tercer ejercicio, tecla 3 presionada.	26
Figura 23.- Simulación en proteus para el tercer ejercicio, tecla # presionada (F).	26
Figura 24.- Prueba 1 del ejercicio 3, tecla 3 presionada.	27
Figura 25.- Prueba 2 del ejercicio 3, tecla # presionada (F).	27

1.-INTRODUCCIÓN

Esta práctica está conformada por tres ejercicios de los cuales el primero de ellos tiene la finalidad de crear cuatro secuencias diferentes en sistema binario que serán mostrados en leds a diferentes frecuencias. Cada una de las secuencias podrá ser seleccionada así como también se podrán seleccionar dos frecuencias a las cuales se realizará cada una de las secuencias

El segundo ejercicio tiene la finalidad de crear un segundero de 0 a 60 que será desplegado en 2 displays de 7 segmentos.

Para el caso del tercer ejercicio se creó un programa para que el microcontrolador fuera capaz de obtener los datos que serán introducidos por un teclado matricial y poder mostrar estos datos a través de leds mostrando en conjunto el valor en sistema binario de la tecla presionada por el usuario.

2.- OBJETIVOS

2.1.- GENERAL

1. Que el alumno conozca el uso de los ciclos de reloj para poder obtener diferentes frecuencias a la salida de los datos así como también aprenda a manejar la pila del microcontrolador ATmega328P.

2.2.- PARTICULARES

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

3.- MARCO TEÓRICO

3.1.- FRECUENCIA DER RELOJ

La frecuencia de reloj en relación a un procesador o microprocesador indica la frecuencia a la cual los transistores que lo conforman conmutan eléctricamente, es decir, abren y cierran el flujo de una corriente eléctrica. La frecuencia es una magnitud física, cuya unidad es el Hertz (Hz) que representa un ciclo u oscilación por segundo, en el caso de los procesadores indica las conmutaciones eléctricas acaecidas en un segundo dentro de un transistor tomado como muestra. Por ejemplo, si un procesador es especificado con una frecuencia de reloj máxima de 2.1 GHz, los transistores que lo componen estarán en la capacidad de conmutar el flujo de una corriente 2.1×10^9 veces por segundo.

Se debe tener presente que el principio de conmutación entre dos estados o lógica binaria aplicado a la electrónica, conjuntamente con el álgebra de Boole hacen posible la electrónica de sistemas digitales, siendo el computador digital una implementación práctica de estos conceptos.

No debe confundirse la frecuencia de reloj con la cantidad de operaciones que un procesador es capaz de realizar en un segundo, puesto que una simple operación, puede demandar como mínimo una

instrucción y esta instrucción demandará al procesador dependiendo de su arquitectura una cierta cantidad de operaciones lógicas que conlleven otra cantidad de conmutaciones electrónicas por parte del arreglo de transistores que conforman el procesador.

3.2.- STACK O PILA

Stack o pila es una estructura de datos con acceso del tipo LIFO (Last In First Out), último en entrar, primero en salir.

Este tipo de estructura de datos es fácil de desarrollar por los sistemas microprocesadores y resulta de gran utilidad para trabajar con listas de datos y es imprescindible para el trabajo interno del microprocesador en las subrutinas e interrupciones.

Algunos microprocesadores pueden tener el Stack en su interior, representando un sistema muy rápido pero de tamaño limitado, la mayoría de microprocesadores disponen el stack en la memoria externa, hecho que proporciona una gran capacidad de almacenamiento y el control de su ubicación, aunque el acceso sea más lento.

3.2.1.- PUNTERO A LA PILA (STACK POINTER)

El puntero del stack SP (stack pointer) como su propio nombre indica es un registro apuntador a la posición de memoria donde se encuentra la pila.

3.2.2.- PUSH

La instrucción PUSH, sitúa en la pila el contenido de la dirección indicada, primero se incrementa el puntero de la pila y acto seguido el contenido de la dirección se copia en la RAM que indica el apuntador de la pila SP.

$$SP = SP + 1$$

$$SP = \text{Dirección}$$

3.2.3.- POP

La instrucción POP, lee el contenido de la dirección de la pila que indica el SP (apuntador del stack) y lo sitúa en la dirección especificada, después decreenta el apuntador de la pila SP dejándolo en la posición anterior.

$$\text{Dirección} = SP$$

$$Sp = Sp - 1$$

4.- MATERIAL

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

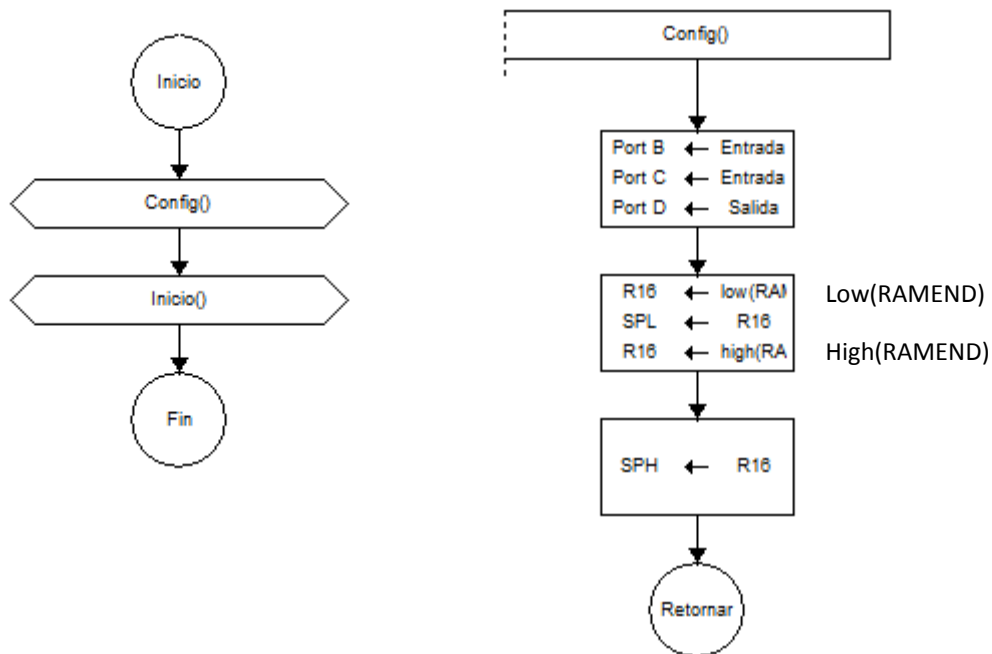
5.- DESARROLLO EXPERIMENTAL

5.1.- SECUENCIAS BINARIAS

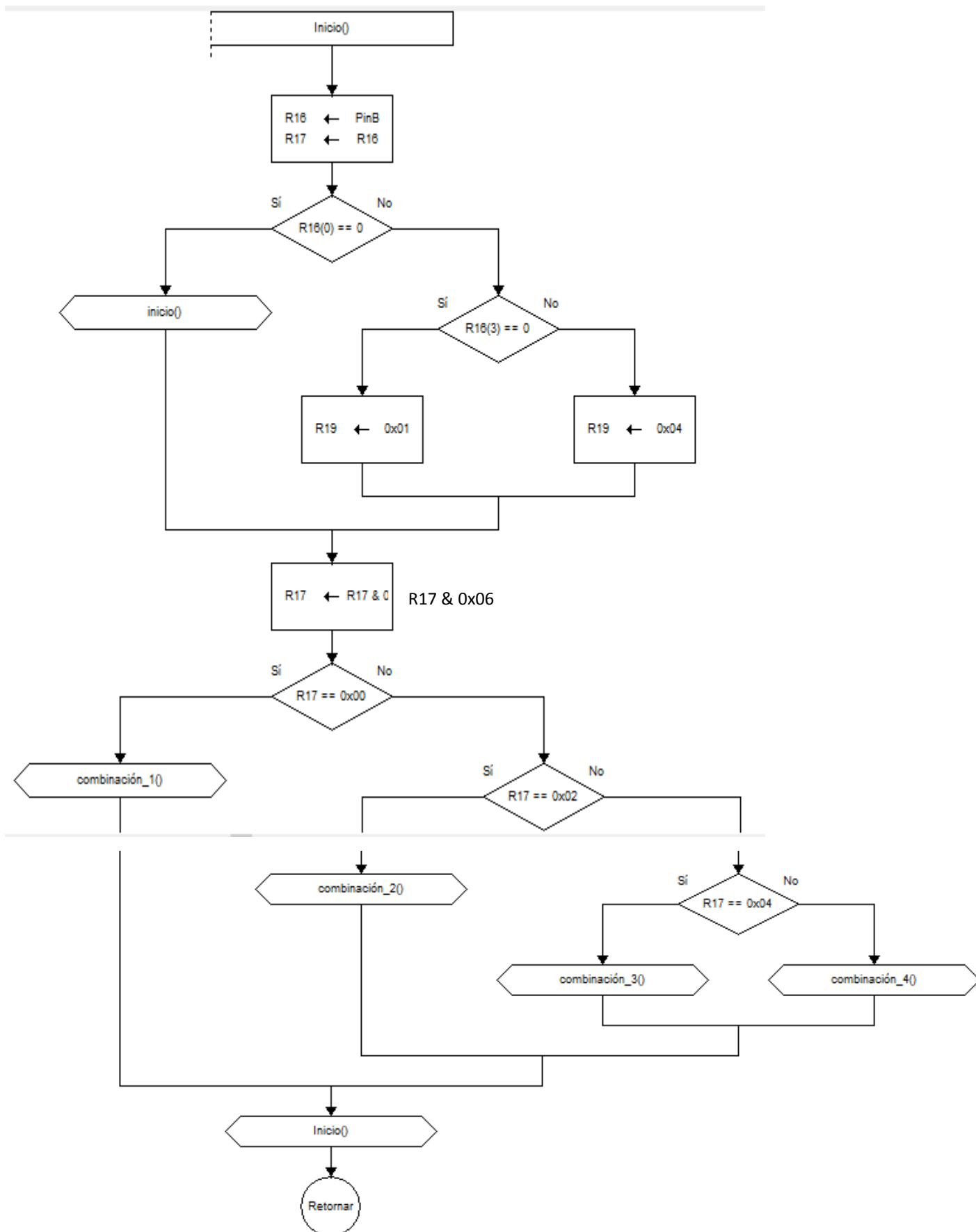
5.1.1.- DIAGRAMA DE FLUJO

A continuación se presenta el diagrama de flujo correspondiente al primer ejercicio de la práctica uno.

Aquí se presenta la subrutina config en donde se configura la pila además del programa principal.

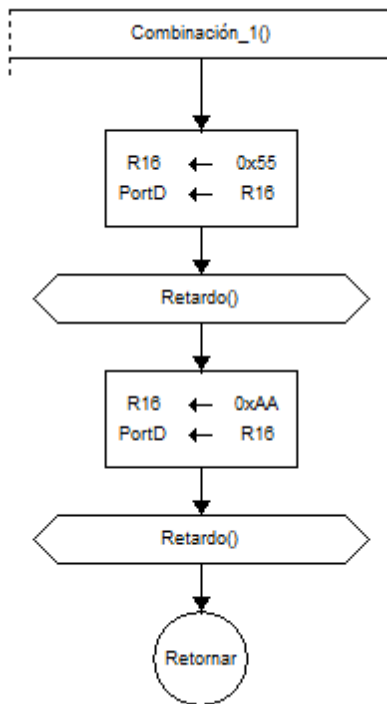


A continuación se presenta la subrutina inicio.

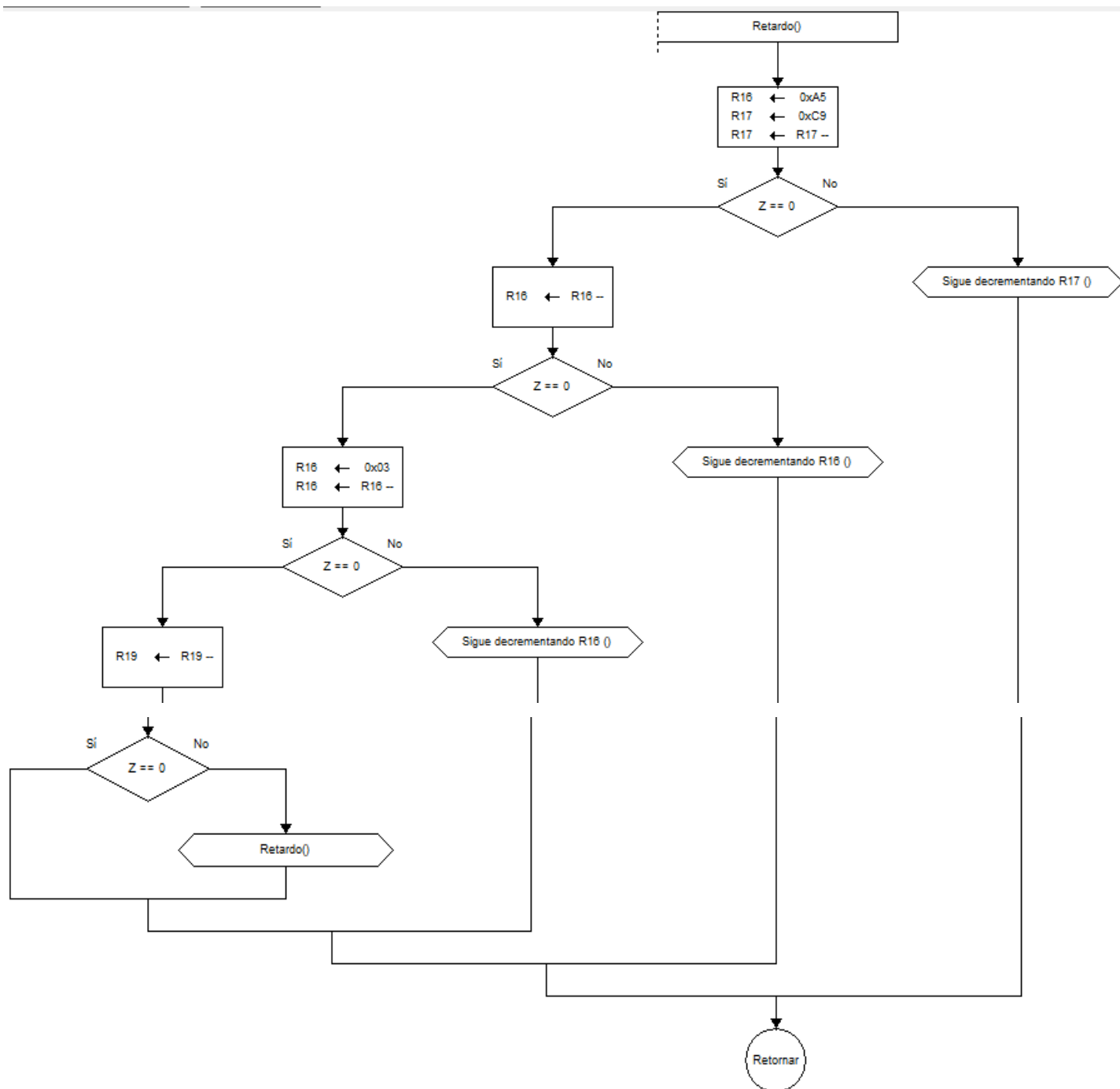


A continuación se presenta la subrutina de combinación_1 para ejemplificar como se manda llamar el retardo para lograr la frecuencia deseada.

Para todas las demás rutinas se realiza el mismo procedimiento solo que varía en el número de secuencias que hay que mostrar por lo que solo se ejemplifica esta combinación.



Como se observa, en esta subrutina se manda llamar a otra subrutina llamada retardo en la cual se lleva a cabo en sí el control de la velocidad. Para poder generar las velocidades solicitadas enviamos un registro que tendrá el valor de 1 y 4, el 1 representa la velocidad de 100 ms y el 4 la de 400 ms y expresará las veces que se repite el código del retardo para evitar hacer dos retardos diferentes puesto que si tiene el valor de 1, el retardo será de 100 ms y si tiene el valor de 4 entonces será 100×4 ms es decir 400 ms.



Por el software utilizado para crear los diagramas de flujo fue imposible hacer que las flechas de las condiciones regresaran a pasos anteriores por lo que se colocaron etiquetas en las cuales se explica que operaciones se ejecutan en cada caso así por ejemplo se agregó una etiqueta que dice sigue decrementando R17 y después se liga al final de la subrutina pero lo que se trató de representar es que el programa sigue decrementando R17 hasta que se encienda la bandera Z y hasta ese momento pasa a la siguiente instrucción la cual tiene una estructura similar a esta. El que se encuentre ligada la etiqueta con el fin de la subrutina no quiere decir que ahí termine la subrutina es por eso que se dio la explicación anterior para evitar esta confusión.

Esas fueron todas las rutinas y subrutinas correspondientes a este ejercicio de la práctica 1 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.

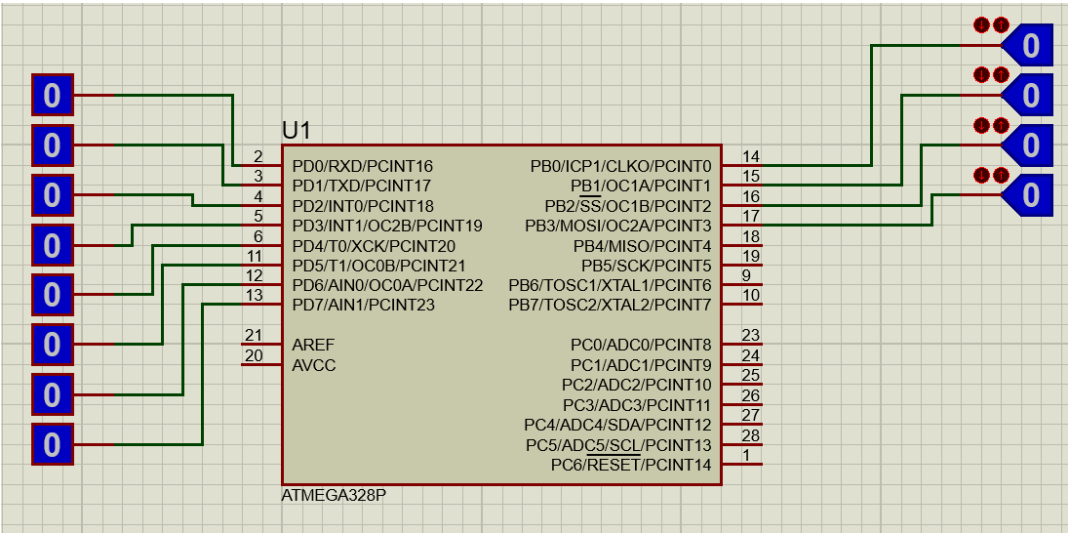


Figura 1.- Circuito a realizar en el laboratorio.

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

5.1.3.- PROCEDIMIENTO

1. Realizar un programa en ensamblador que tenga programadas 4 secuencias de 8 bits y pueda tener la opción de ponerlo a dos velocidades de cambio distintas (400 ms y 100 ms), en el diseño incluir un botón de on/off. Utilizar rutinas de tiempo y rutinas para cada secuencia.

A continuación se presenta una tabla en la que se muestran las rutinas a reproducir con la ayuda del microcontrolador.

	1000 0000	0000 1000
0101 0101	0100 0000	0000 0100
1010 1010	0010 0000	0000 0010
	0001 0000	0000 0001
1100 1100	1000 0001	0010 0100
0110 0110	0100 0010	0100 0010
0011 0011	0010 0100	
1001 1001	0001 1000	

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

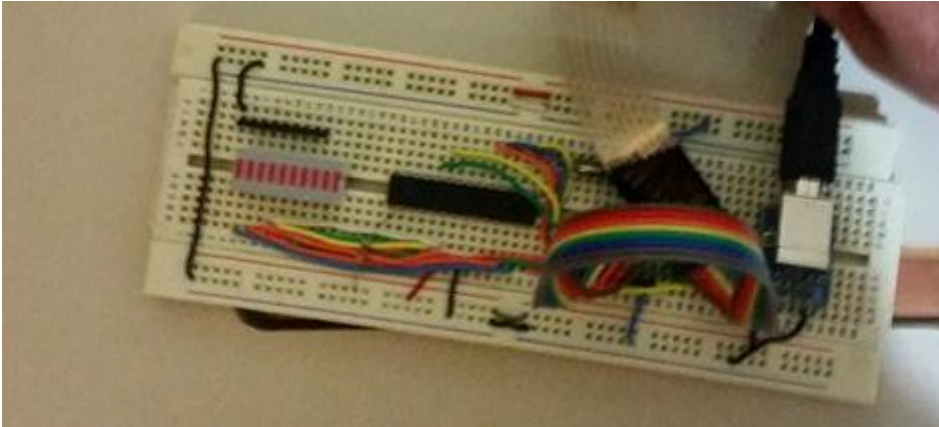


Figura 2.- Circuito correspondiente al primer ejercicio de la práctica 1.

Una vez armado el circuito de la **figura 5** 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_1A - AtmelStudio
File Edit View Project Build Debug Tools Help
Practica_1A.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

config:
clr R16; Limpia el registro 16.
out DDRB, R16; Configura el puerto B como entrada.
out DDRC, R16; Configura el puerto C como entrada.
ser R16;Pone en alto todo el registro.
out DDRD, R16; Configura el puerto D como salida.
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.

inicio:
;RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0;
;-- -- -- -- Vel Comb On/Off.
clr R16; Limpia R16.
out PortD,R16; Pone en cero todas las salidas.
in R16, PinB;Manda las entradas del puerto B a R16.
mov R17, R16;Copia el contenido de R16 a R17.
sbrs R16,0; Verifica que RB0 sea 0.
jmp inicio; Esta apagado.
sbrs R16,3; Verifica que RB3 sea 0
jmp Carga_100;
jmp Carga_400;
```

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.

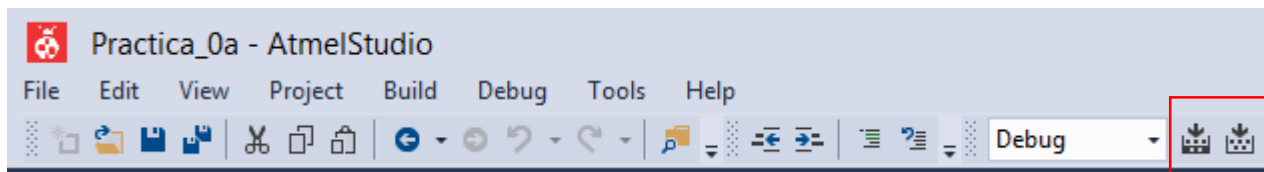


Figura 4.- Localización de la opción construir.

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 1 tal como se muestra en la **figura 5**.

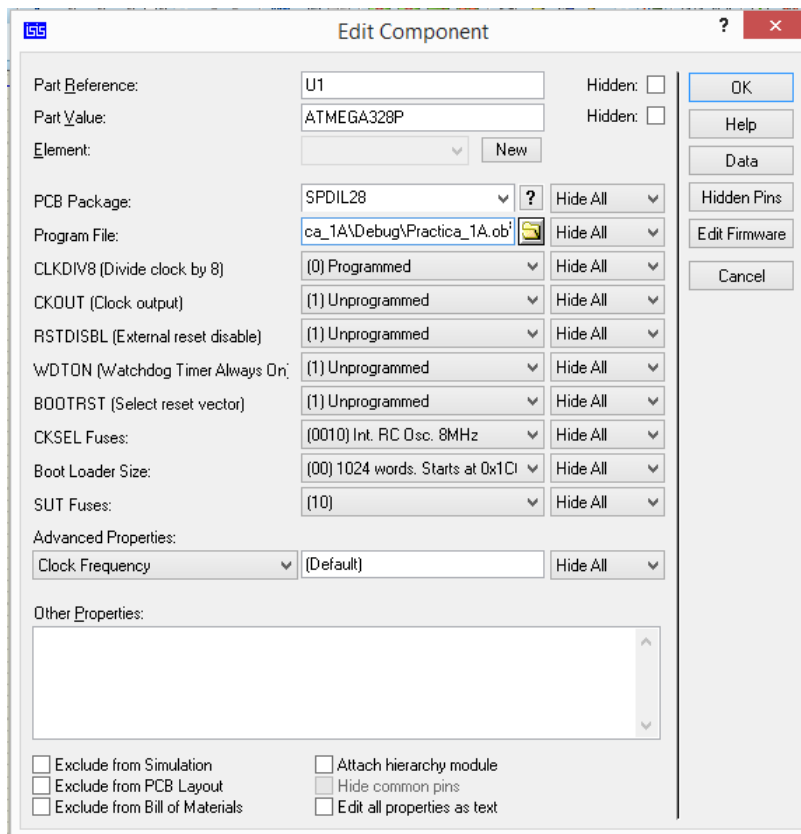


Figura 5.- Selección del archivo .hex para simularlo en proteus.

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.

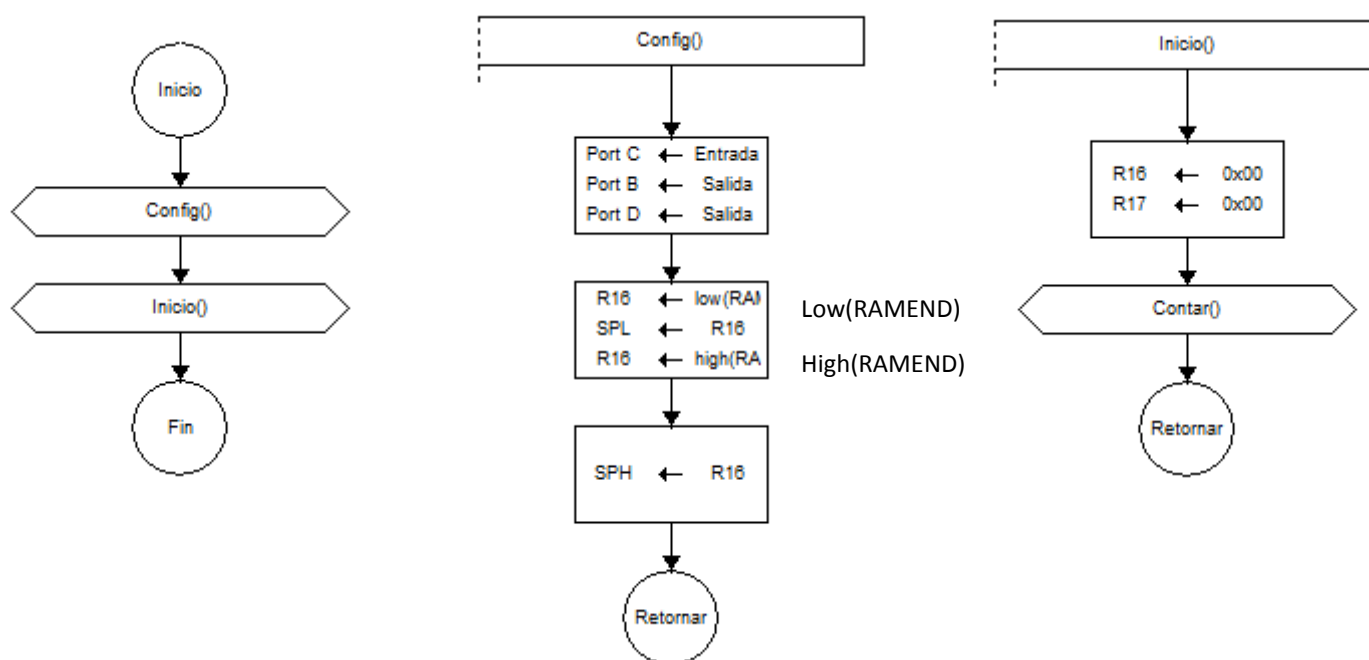
Los resultados obtenidos en este primer ejercicio de la práctica están documentados en la sección de resultados en su apartado correspondiente para el primer ejercicio.

5.2.- SEGUNDERO (0 - 60)

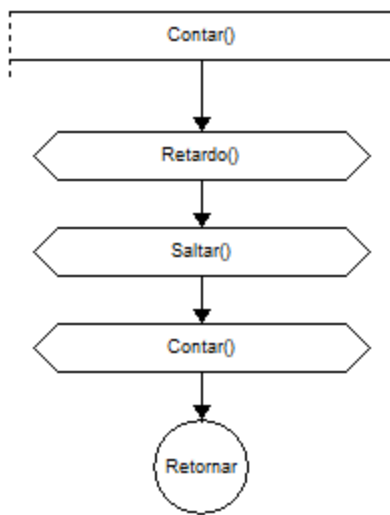
5.2.1.- DIAGRAMA DE FLUJO

A continuación se presenta el diagrama de flujo correspondiente al segundo ejercicio de la práctica 1.

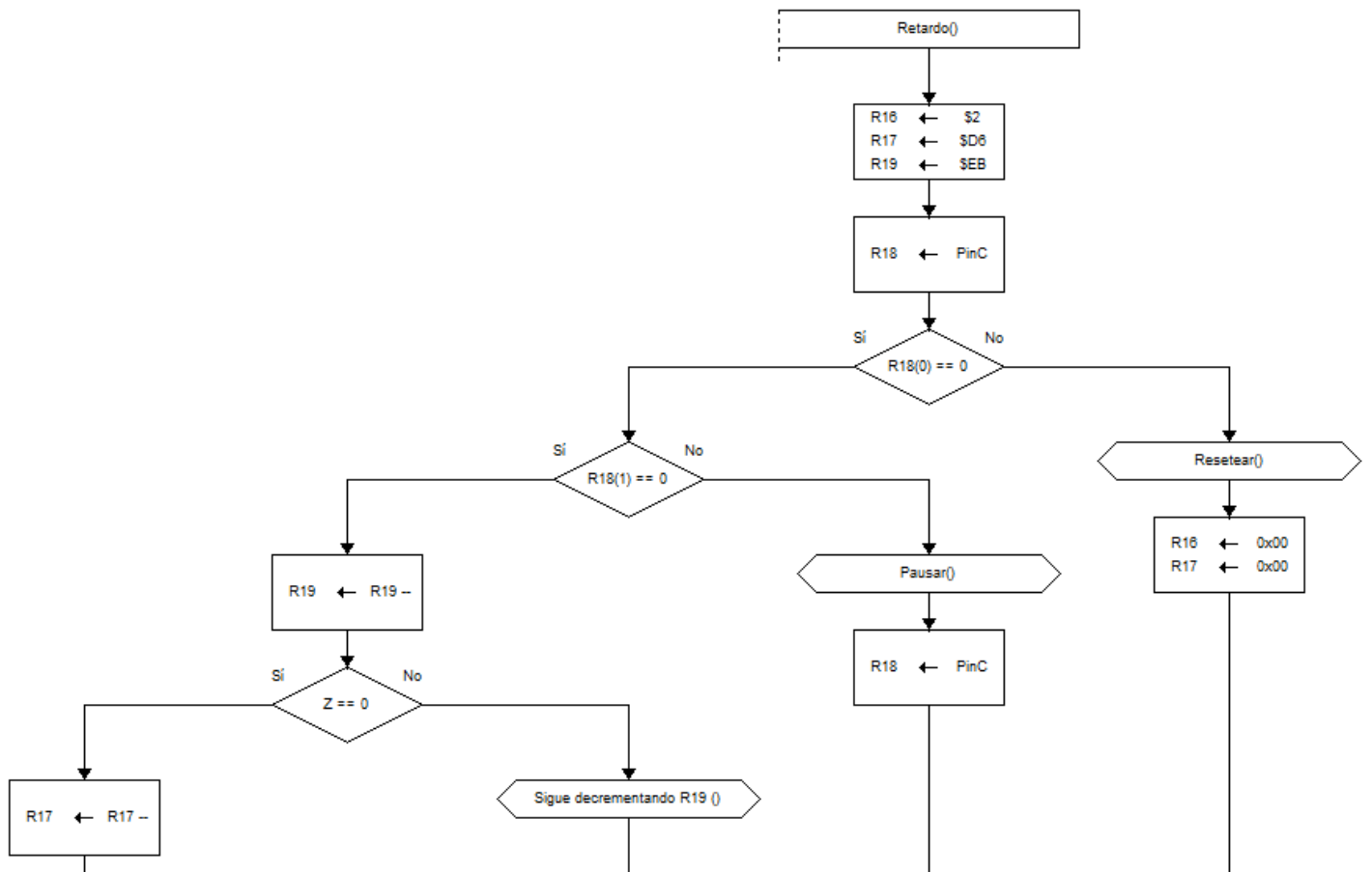
Aquí se presenta la subrutina config en donde se configura la pila además del programa principal además de la subrutina inicio.

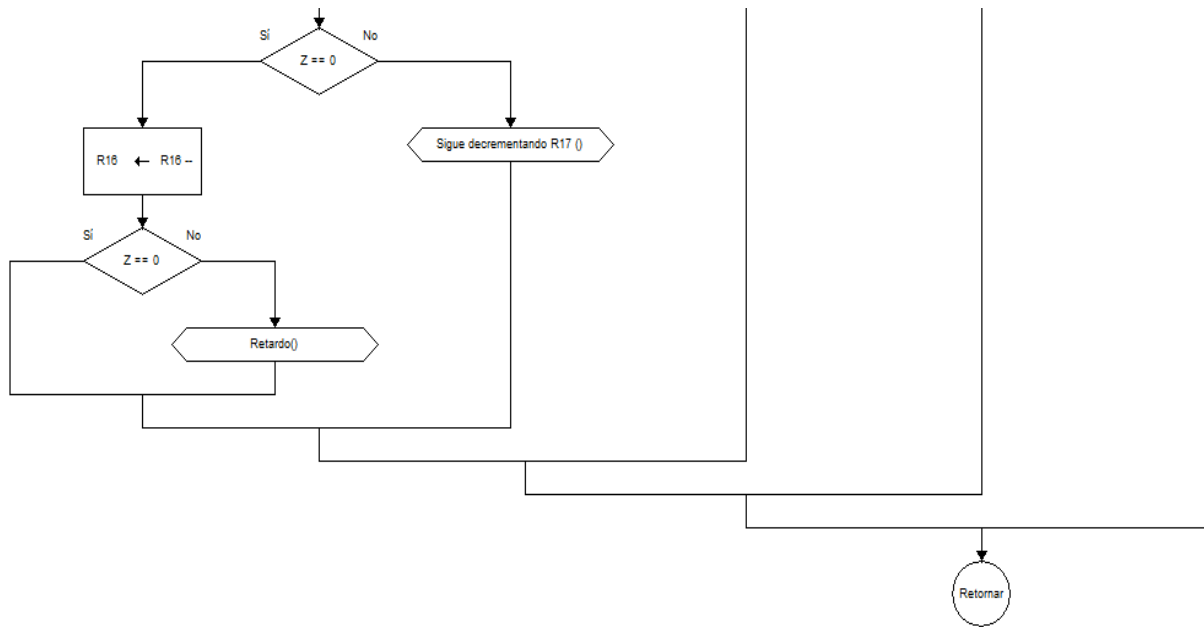


En las rutinas anteriores se presentan otras subrutinas las cuales se colocan a continuación y como se puede observar la rutina principal solo tiene dos subrutinas.



La subrutina contar contiene tres subrutinas que son las que en sí contienen la mayor cantidad de código y por lo tanto las que rigen el comportamiento del microcontrolador.

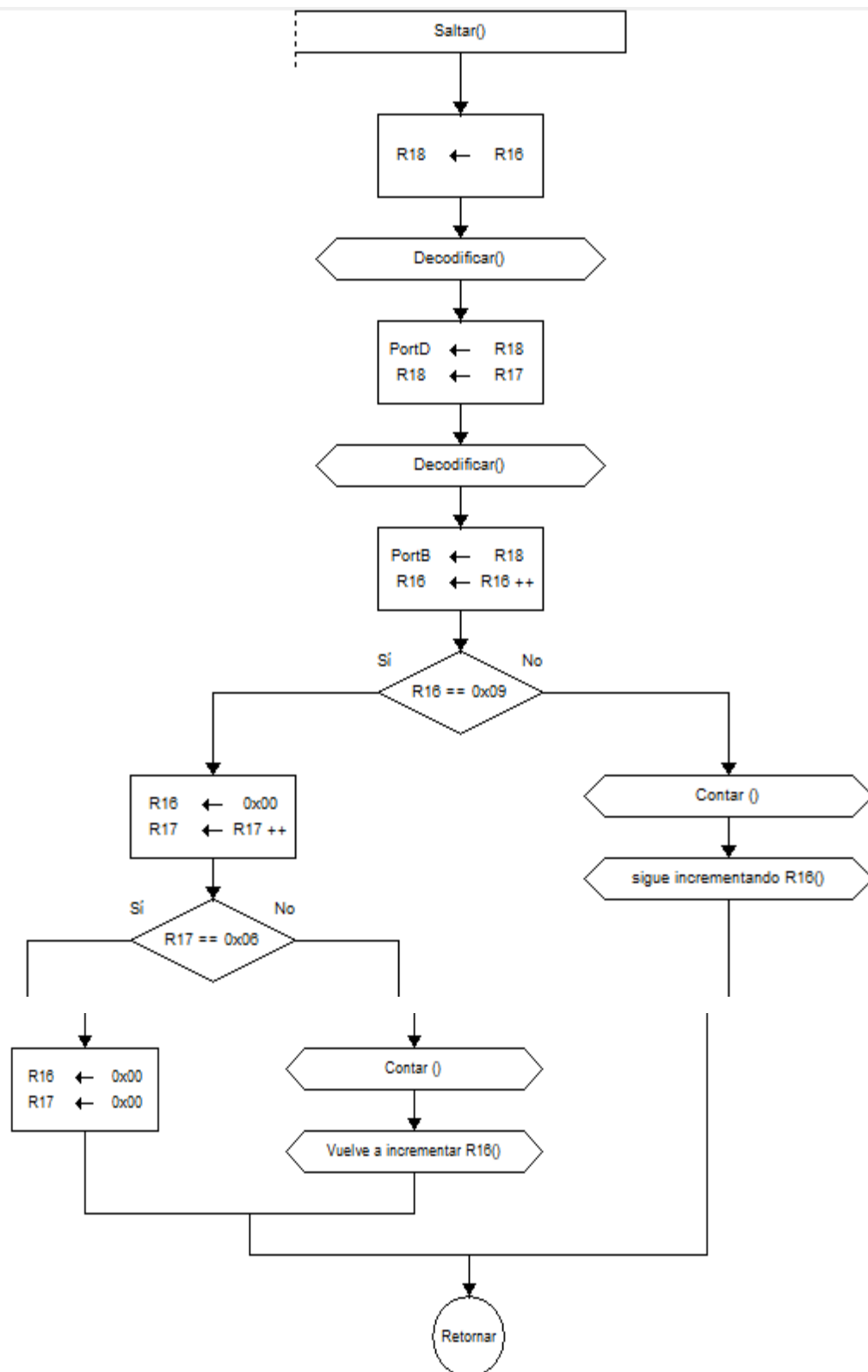




En la subrutina retardo se observa que es aquí es donde se toma en cuenta si el usuario decide resetear o poner en pausa el programa. Básicamente el retardo lo que hace es quedarse ejecutando código hasta que se cumpla una cierta cantidad de ciclos de reloj que dan como resultado un tiempo aproximado de 1 segundo para así aumentar cada segundo el contador a hasta 60.

Se utilizaron etiquetas para indicar el proceso que se debe de seguir ejecutando en caso de que las comparaciones sean falsas, esto para las primeras 4 comparaciones por que la última comparación si es correcta ya que en el caso de que sea un resultado positivo la subrutina termina y si es un resultado negativo entonces la subrutina se vuelve a ejecutar a si misma.

A continuación se presenta el subprograma encargado de incrementar los contadores y de enviar el resultado a la salida para ser mostrado por los displays.



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.

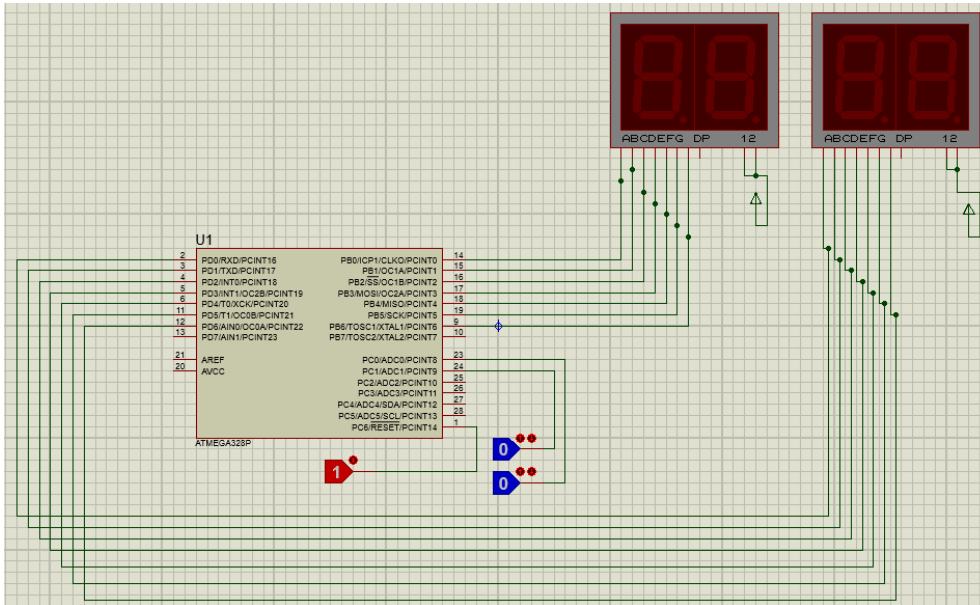


Figura 6.- Circuito a realizar en el laboratorio correspondiente al segundo ejercicio.

Como se observa en la **figura 6** se utilizaron en la simulación probadores y estados lógicos para simplificar el circuito simulado pero en la práctica se conectaron displays de 7 segmentos de ánodo común y DIP switches para observar el comportamiento de este segundo ejercicio de la práctica 1.

5.2.3.- PROCEDIMIENTO

1. Elaborar un segundero que cuente de 0 a 59 y lo muestre en dos displays de 7 segmentos de ánodo común que va transcurriendo. Habilitar un botón de reset y un botón de pausa.

En la siguiente imagen se puede observar la configuración de este tipo de displays.

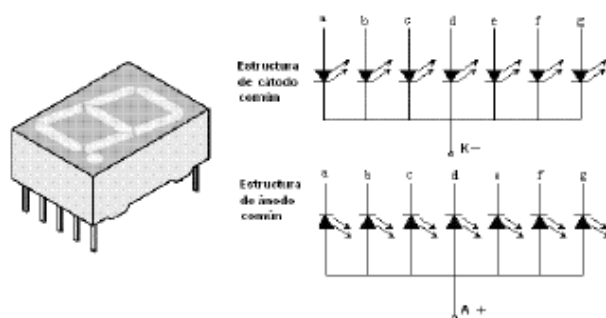


Figura 7.- Configuración de un display de 7 segmentos.

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

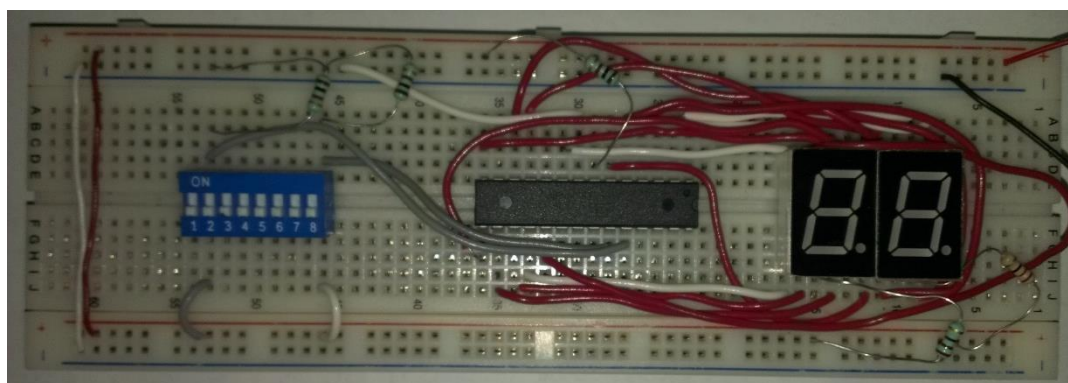
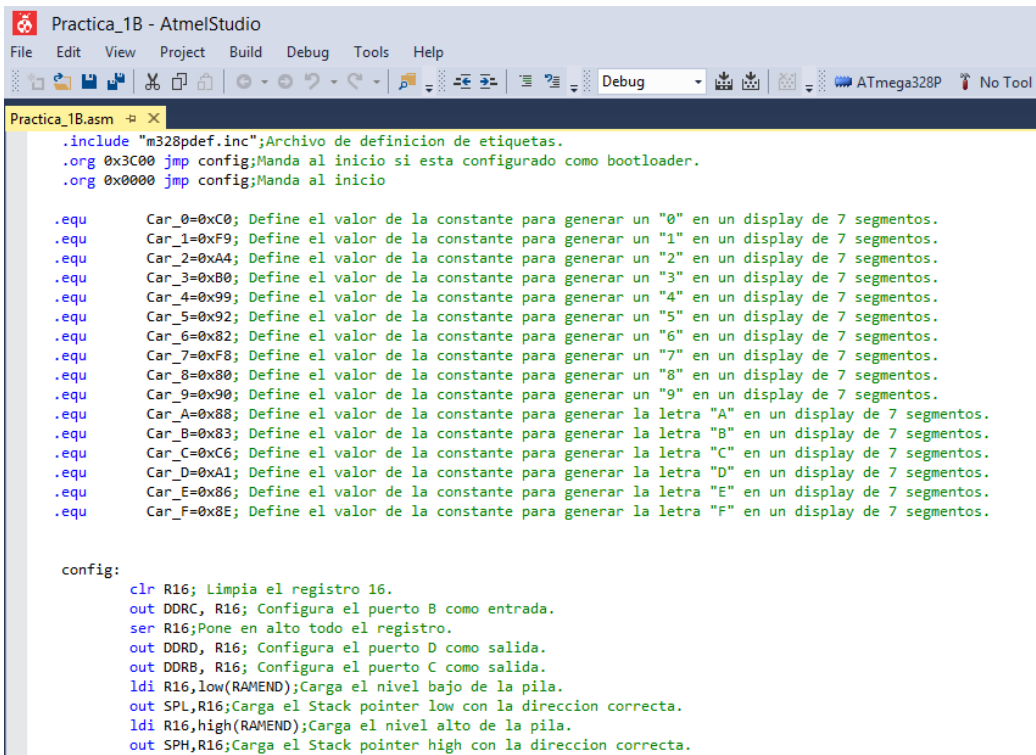


Figura 8.- Circuito correspondiente al segundo ejercicio de la práctica 1.

Como se puede observar en la **figura 8** se cuenta con dos displays de 7 segmentos, uno para las decenas y otro para las unidades además también cuenta con DIP switches que nos sirven para poder poner en pausa el conteo o par resetearlo.

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_1B 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

.equ      Car_0=0xC0; Define el valor de la constante para generar un "0" en un display de 7 segmentos.
.equ      Car_1=0xF9; Define el valor de la constante para generar un "1" en un display de 7 segmentos.
.equ      Car_2=0xA4; Define el valor de la constante para generar un "2" en un display de 7 segmentos.
.equ      Car_3=0xB0; Define el valor de la constante para generar un "3" en un display de 7 segmentos.
.equ      Car_4=0x99; Define el valor de la constante para generar un "4" en un display de 7 segmentos.
.equ      Car_5=0x92; Define el valor de la constante para generar un "5" en un display de 7 segmentos.
.equ      Car_6=0x88; Define el valor de la constante para generar un "6" en un display de 7 segmentos.
.equ      Car_7=0xF8; Define el valor de la constante para generar un "7" en un display de 7 segmentos.
.equ      Car_8=0x80; Define el valor de la constante para generar un "8" en un display de 7 segmentos.
.equ      Car_9=0x90; Define el valor de la constante para generar un "9" en un display de 7 segmentos.
.equ      Car_A=0x88; Define el valor de la constante para generar la letra "A" en un display de 7 segmentos.
.equ      Car_B=0x83; Define el valor de la constante para generar la letra "B" en un display de 7 segmentos.
.equ      Car_C=0xC6; Define el valor de la constante para generar la letra "C" en un display de 7 segmentos.
.equ      Car_D=0xA1; Define el valor de la constante para generar la letra "D" en un display de 7 segmentos.
.equ      Car_E=0x86; Define el valor de la constante para generar la letra "E" en un display de 7 segmentos.
.equ      Car_F=0x8F; Define el valor de la constante para generar la letra "F" en un display de 7 segmentos.

config:
    clr R16; Limpia el registro 16.
    out DDRC, R16; Configura el puerto B como entrada.
    ser R16;Pone en alto todo el registro.
    out DDRD, R16; Configura el puerto D como salida.
    out DDRB, R16; Configura el puerto C como salida.
    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.
```

Figura 9.- 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.

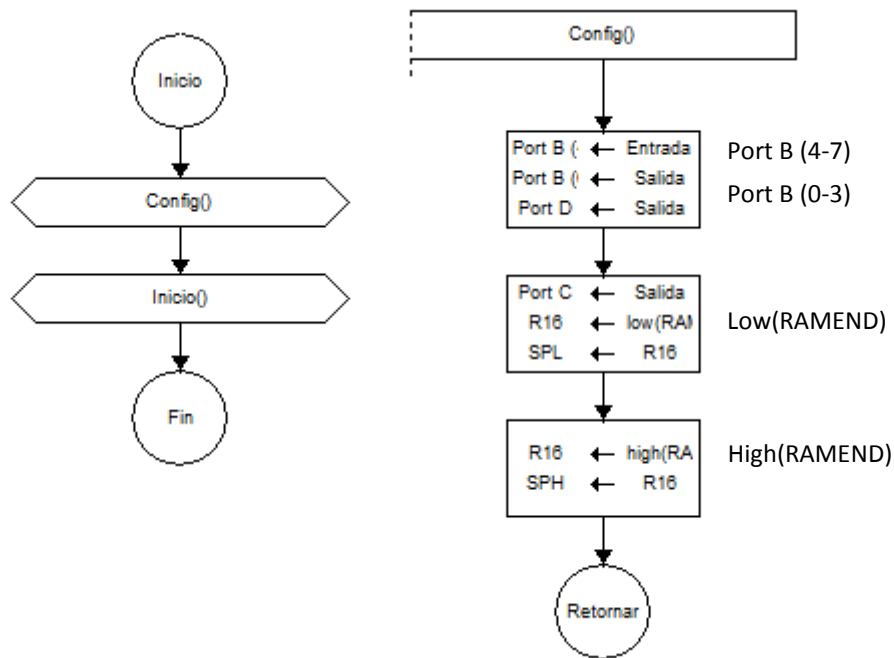
Los resultados obtenidos en este ejercicio se encuentran en su respectivo apartado en la sección de resultados.

5.3.- TECLADO MATRICIAL 4x4

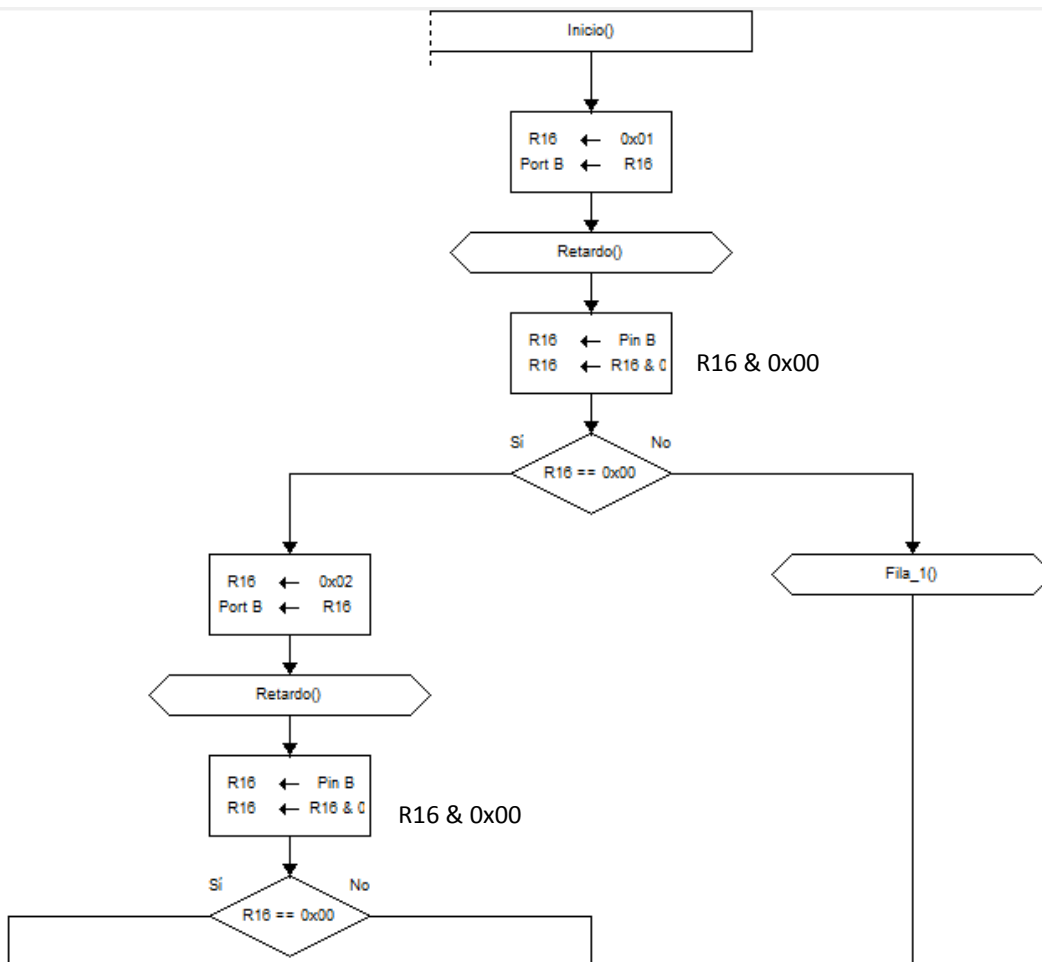
5.3.1.- DIAGRAMA DE FLUJO

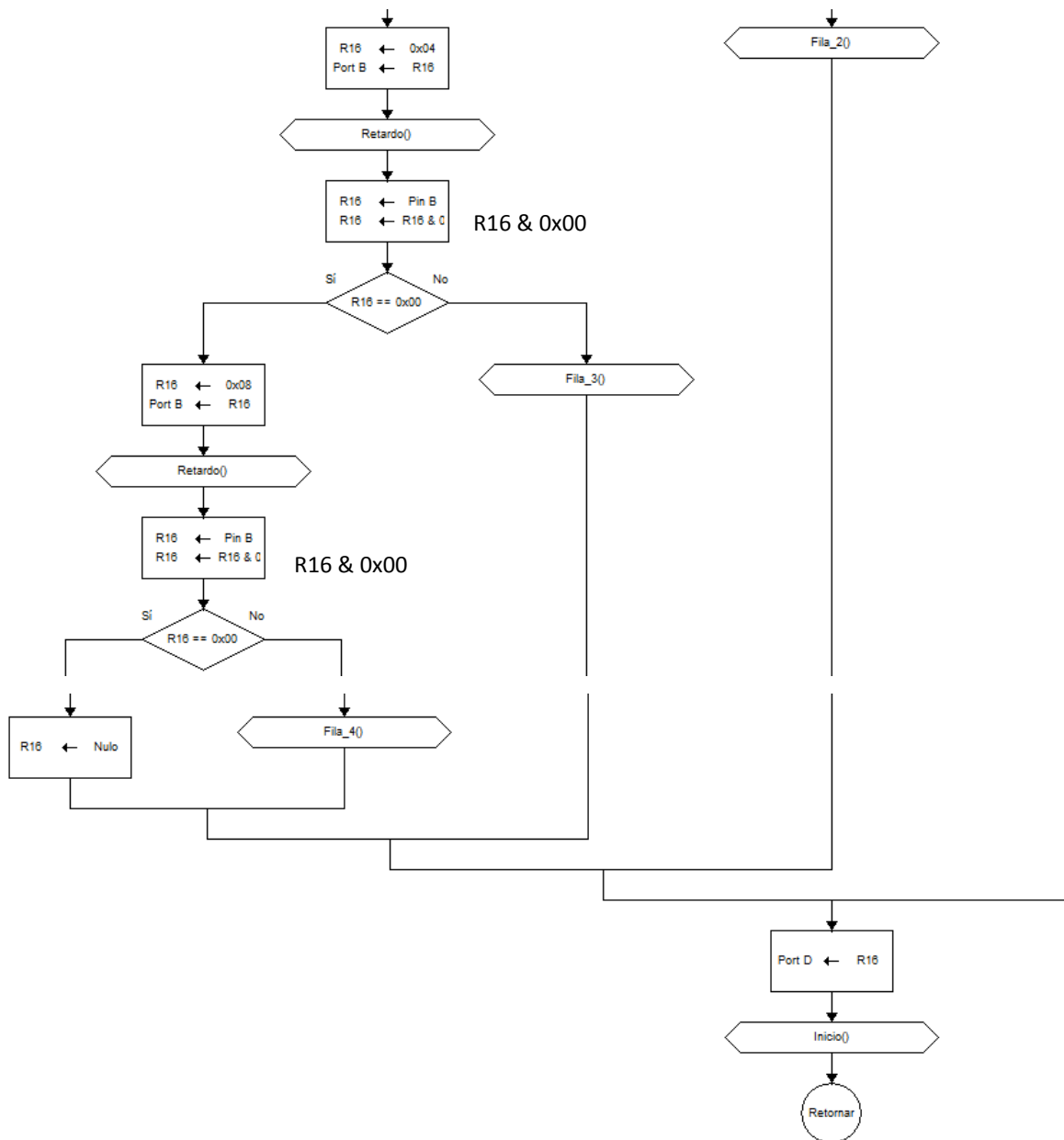
A continuación se presenta el diagrama de flujo correspondiente al tercer ejercicio de la práctica 1.

En el caso de este ejercicio también es necesario realizar la configuración de la pila para poder utilizarla además de la configuración de puertos de entrada y de salida lo que se realiza en la subrutina config tal y como se muestra a continuación.



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



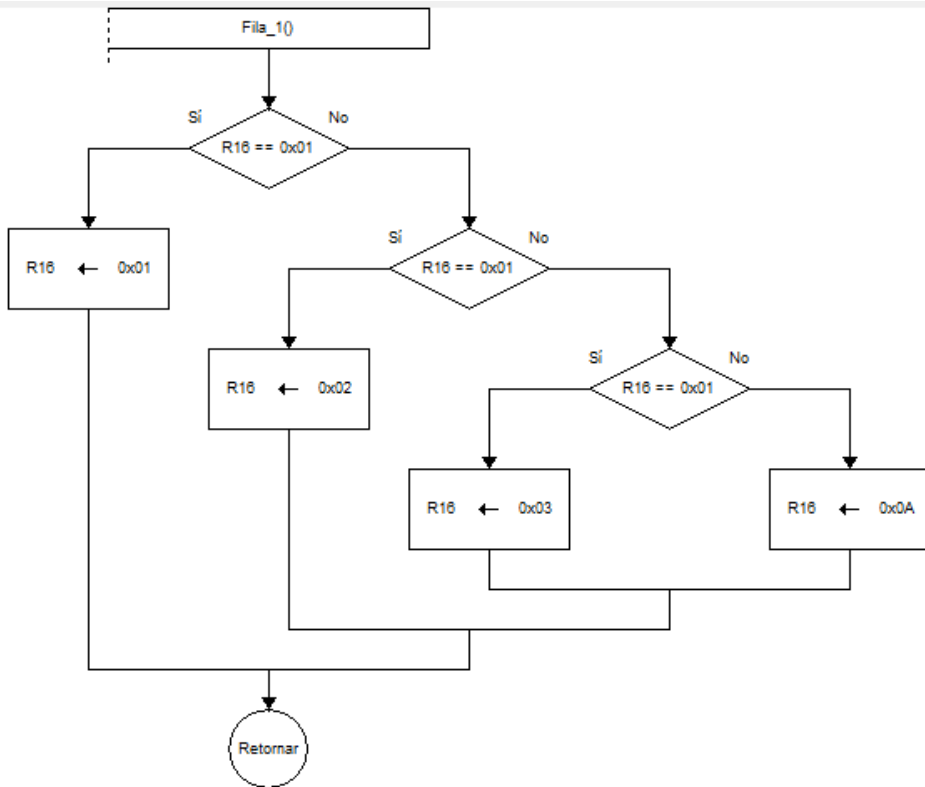


En la rutina anterior se encuentran algunos valores que no fue posible visualizar dentro de los cuadros de operación sin embargo se agregaron cuadros de texto para hacer la aclaración correspondiente del valor que debe de tener.

Todas y cada una de las subrutinas de fila 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, R16 toma un valor el cual al regresar a la subrutina inicio será mostrado por el puerto D.



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.

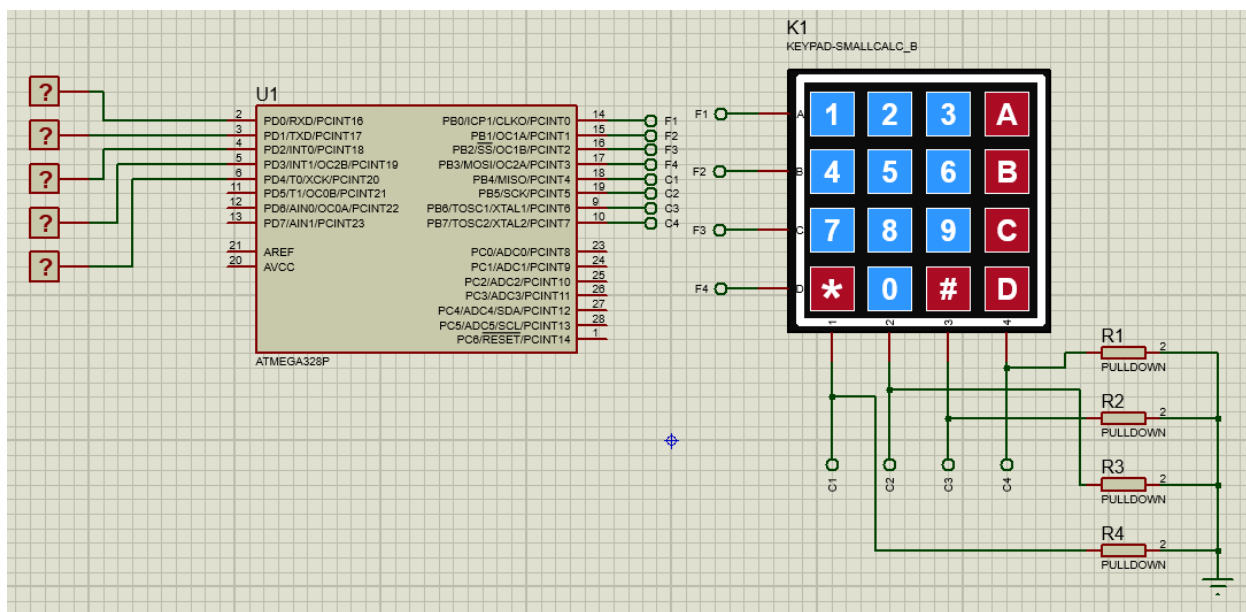


Figura 10.- Circuito a realizar en el laboratorio correspondiente al tercer ejercicio.

Como se observa en la **figura 10** se utilizaron en la simulación probadores lógicos para simplificar el circuito simulado pero en la práctica se conectaron leds para observar el comportamiento de este tercer ejercicio de la práctica 1.

5.3.3.- PROCEDIMIENTO

1. Realizar un programa en ensamblador para controlar un teclado matricial 4x4, al leer la tecla presionada mostrar la información en leds donde se muestre su valor en sistema binario, incluir un led que esté activado para indicar que no se ha presionado ninguna tecla.

En la siguiente imagen se puede observar la configuración de este tipo de teclado.

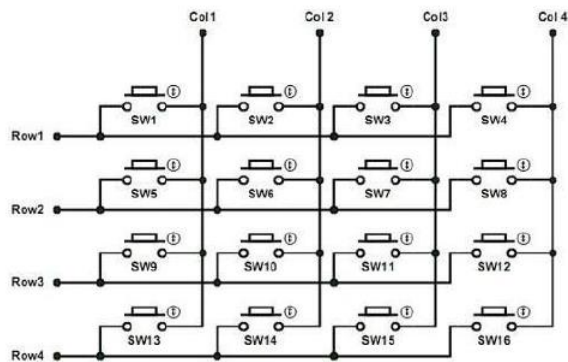


Figura 11.- Configuración de un teclado matricial 4x4.

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

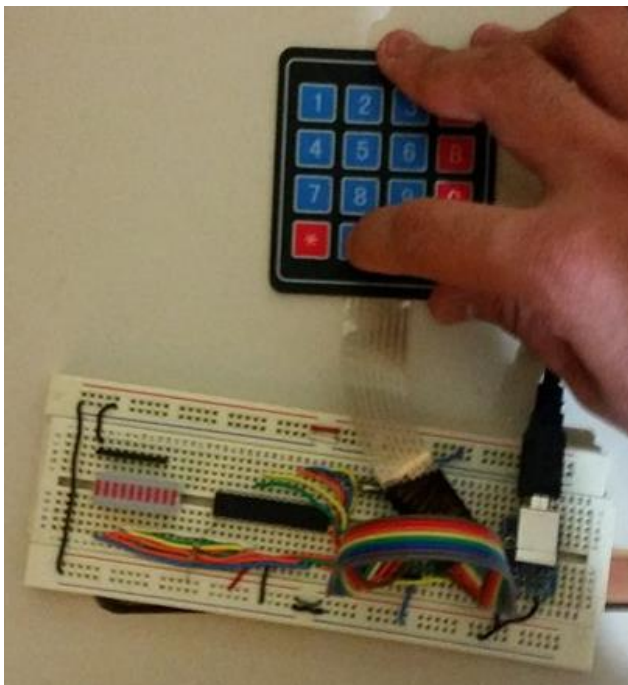
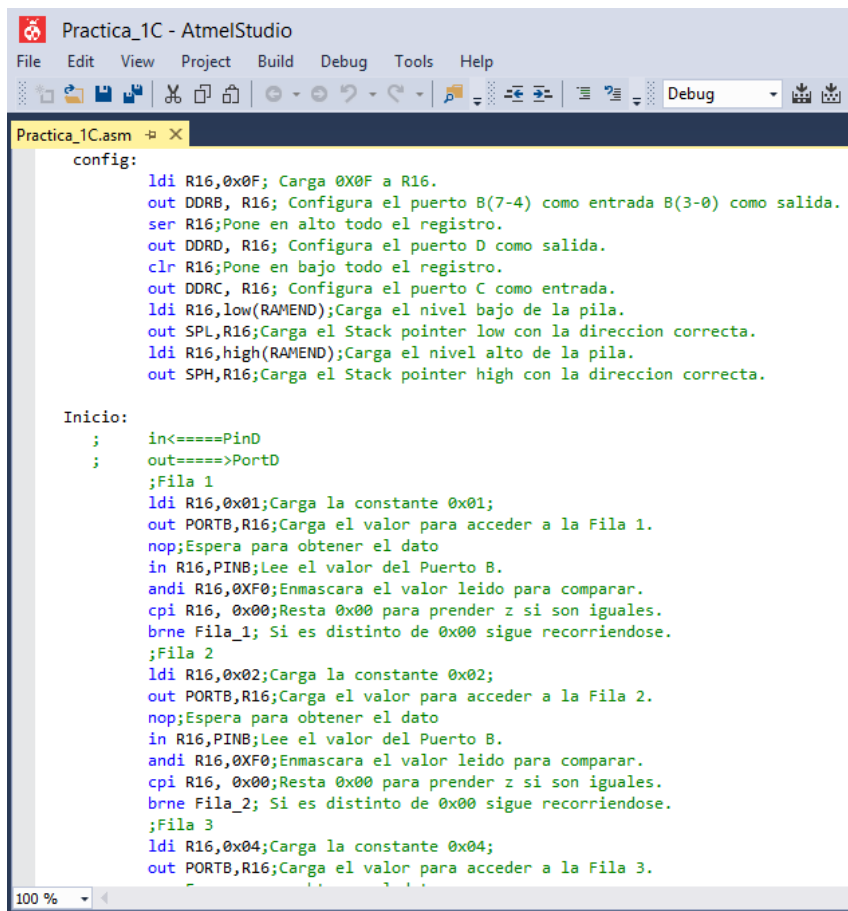


Figura 12.- Circuito correspondiente al tercer ejercicio de la práctica 1.

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



```
Practica_1C - AtmelStudio
File Edit View Project Build Debug Tools Help
Practica_1C.asm
config:
    ldi R16,0x0F; Carga 0x0F a R16.
    out DDRB, R16; Configura el puerto B(7-4) como entrada B(3-0) como salida.
    ser R16;Pone en alto todo el registro.
    out DDRD, R16; Configura el puerto D como salida.
    clr R16;Pone en bajo todo el registro.
    out DDRC, R16; Configura el puerto C 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.

Inicio:
    ; in<====PinD
    ; out====>PortD
    ;Fila 1
    ldi R16,0x01;Carga la constante 0x01;
    out PORTB,R16;Carga el valor para acceder a la Fila 1.
    nop;Espera para obtener el dato
    in R16,PINB;Lee el valor del Puerto B.
    andi R16,0XF0;Enmascara el valor leído para comparar.
    cpi R16, 0x00;Resta 0x00 para prender z si son iguales.
    brne Fila_1; Si es distinto de 0x00 sigue recorriendose.
    ;Fila 2
    ldi R16,0x02;Carga la constante 0x02;
    out PORTB,R16;Carga el valor para acceder a la Fila 2.
    nop;Espera para obtener el dato
    in R16,PINB;Lee el valor del Puerto B.
    andi R16,0XF0;Enmascara el valor leído para comparar.
    cpi R16, 0x00;Resta 0x00 para prender z si son iguales.
    brne Fila_2; Si es distinto de 0x00 sigue recorriendose.
    ;Fila 3
    ldi R16,0x04;Carga la constante 0x04;
    out PORTB,R16;Carga el valor para acceder a la Fila 3.
```

Figura 13.- 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.- SECUENCIAS BINARIAS

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.

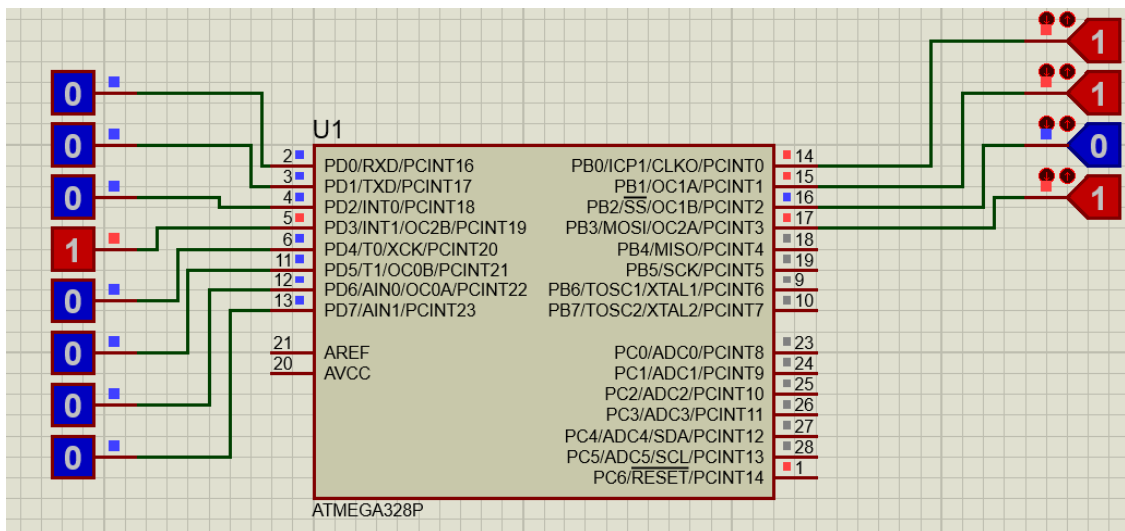


Figura 14.- Simulación en proteus para el primer ejercicio, secuencia 2.

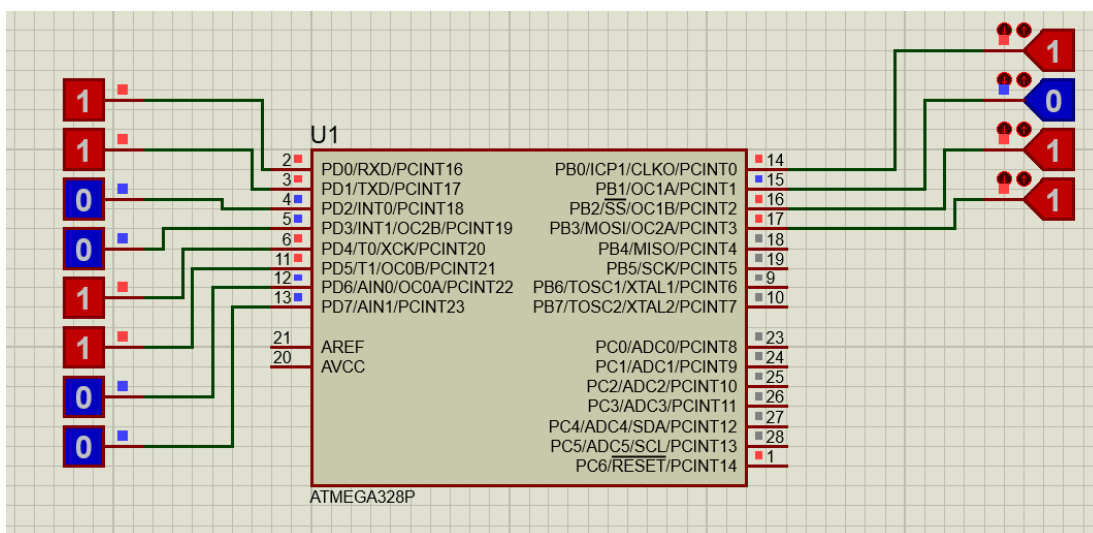


Figura 15.- Simulación en proteus para el primer ejercicio, secuencia 3.

Como se ve en las **figuras 14 y 15** el circuito funcionó correctamente.

En las **figuras 16 y 17** se puede observar el correcto funcionamiento del circuito armado en el laboratorio, estas figuras corresponden con las **figuras 14 y 15** pues son los mismos ejemplos tanto en la simulación como en el circuito armado.

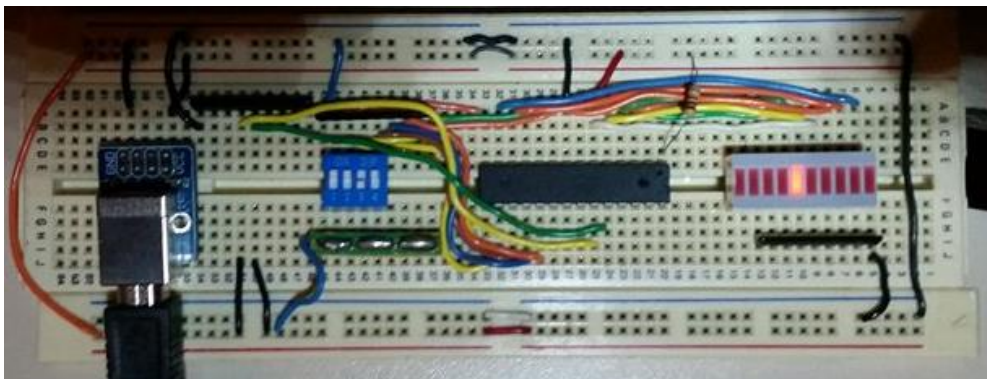


Figura 16.- Prueba 1 del ejercicio 1, secuencia 2.

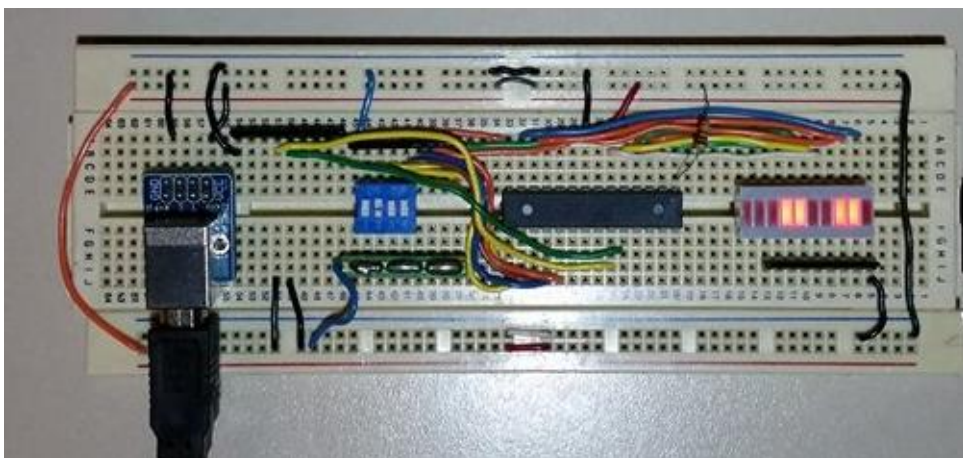


Figura 17.- Prueba 2 del ejercicio 1, secuencia 3.

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

6.2.- SEGUNDERO (0 - 60)

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.

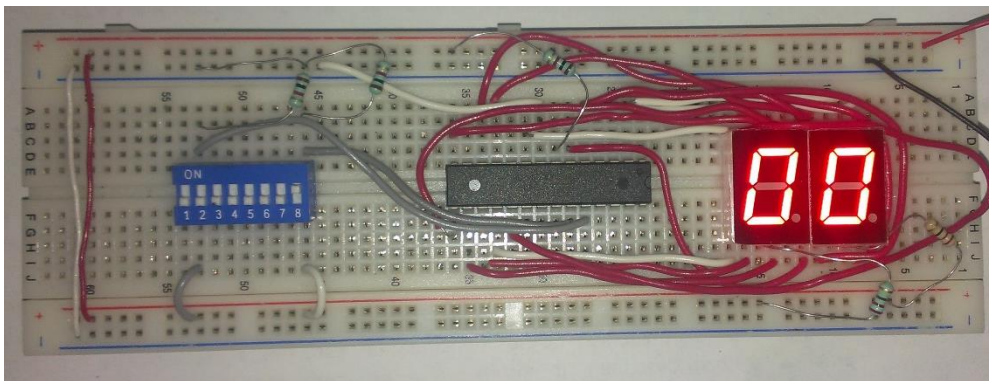


Figura 20.- Prueba 1 del ejercicio 2, reset activado.

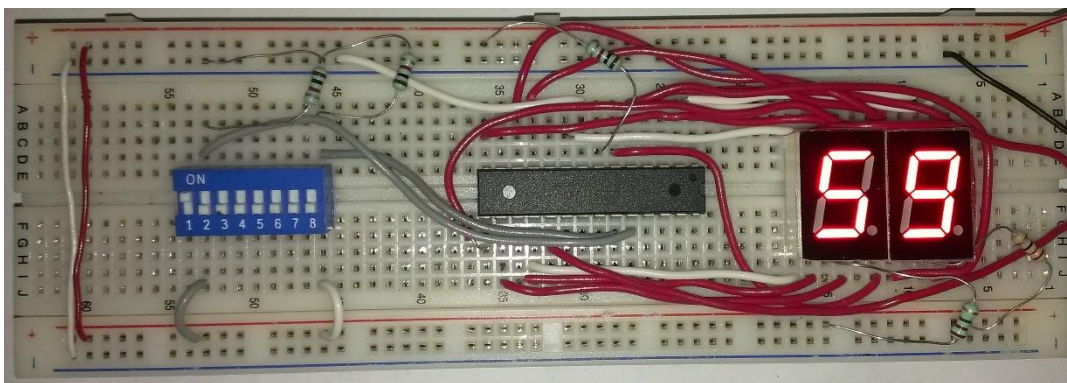


Figura 21.- Prueba 2 del ejercicio 2, pausa activada.

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

6.3.- TECLADO MATRICIAL 4x4

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.

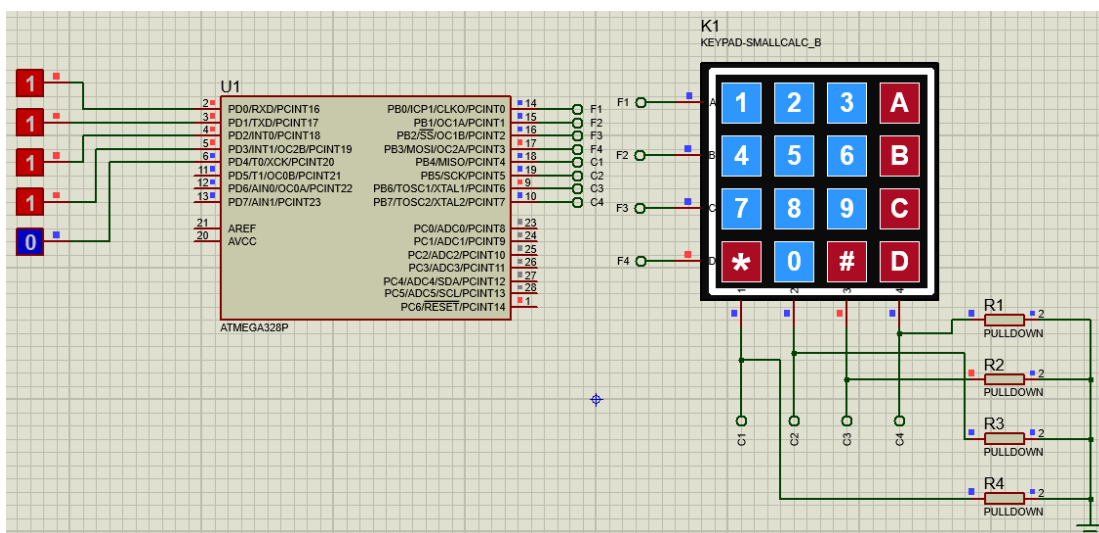
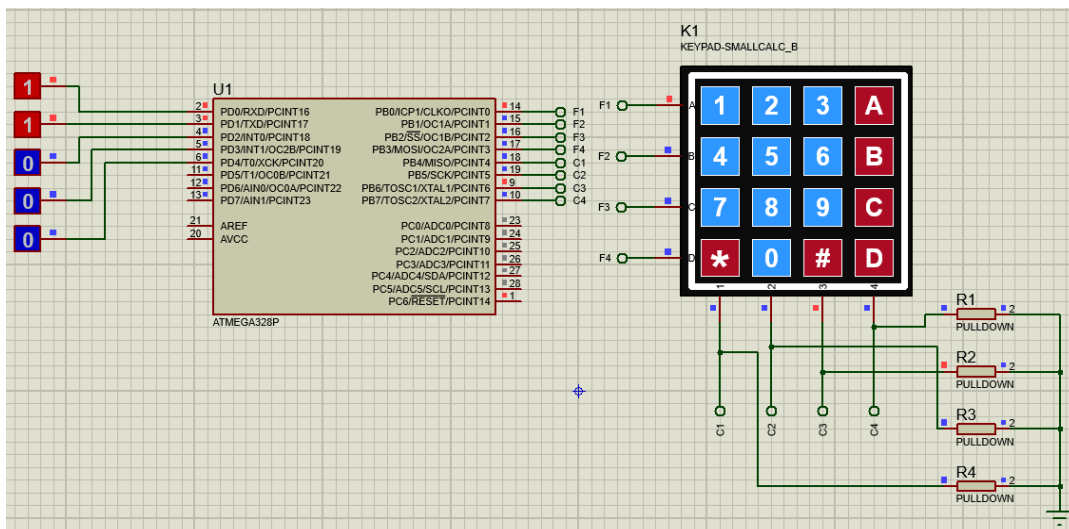


Figura 23.- Simulación en proteus para el tercer ejercicio, tecla # presionada (F).

Como se ve en la **figura 22** y en la **figura 23** el circuito respondió correctamente y esto también se comprueba con las **figuras 24** y **25** en las cuales se observa el mismo resultado sobre el circuito armado en el laboratorio.

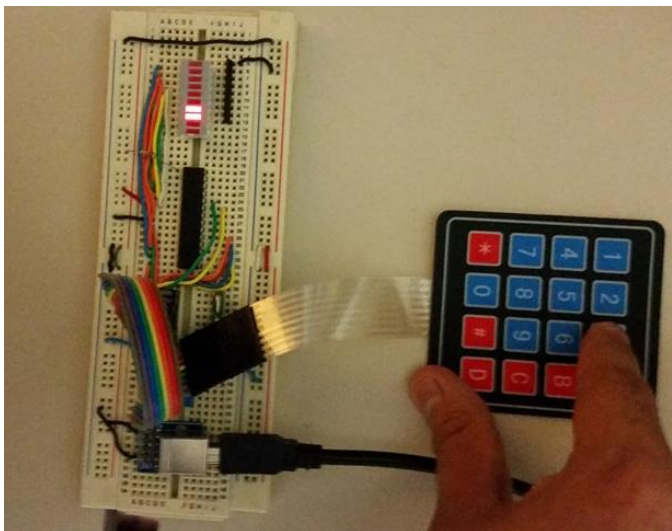


Figura 24.- Prueba 1 del ejercicio 3, tecla 3 presionada.

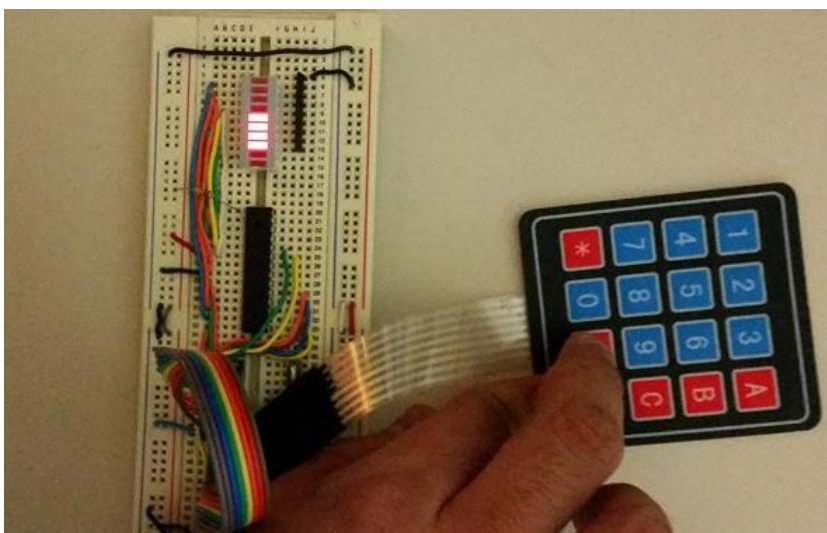


Figura 25.- Prueba 2 del ejercicio 3, tecla # presionada (F).

Con las figuras anteriores se comprueba el correcto funcionamiento y programación del ejercicio 3 de esta práctica.

7.- CONCLUSIONES

- Aguilar Zarazúa Luis Fernando

En la realización de esta práctica logramos comprobar los conceptos sobre las subrutinas y las aplicaciones de las banderas vistas en clase, ya que por medio del microcontrolador podíamos procesar diferentes entradas digitales con las cuales asignamos procesos en determinados momentos siendo así un ejemplo de un circuito secuencial ya que en los 3 casos dependiendo del tiempo mostrábamos la salida correspondiente.

En el caso de las subrutinas son una capacidad del microcontrolador muy útil, ya que beneficia tanto al programador por que no debe de realizar códigos repetitivos al igual que al microcontrolador debido a que puede reutilizar en segmento de memoria de programa para repetir secuencias parecidas. Además de esto al momento de programar el teclado matricial podemos observar que a pesar de ser más hardware este se puede simplificar si se le da un correcto uso secuencial el cual el microcontrolador otorga de una forma muy eficiente y en caso de la secuencia de leds también se puede observar que al tener una ALU y la memoria RAM podemos guardar datos útiles como el tiempo necesario según una condición de entrada.

- Martínez Pérez Nestor

Con la realización de esta práctica se pudieron llevar al laboratorio conceptos vistos en la clase tales como los retardos ya que fueron necesarios para los 3 ejercicios aunque en el tercer ejercicio se utilizó, para generar el retardo, una instrucción nula que su principal función es la de hacer que el microprocesador trabaje pero sin ejecutar ninguna instrucción con lo que se logra un retardo lo suficientemente pequeño para poder sensar los puertos de entrada y enviar los datos por los puertos de salida a una velocidad muy alta para asegurar que no tendremos pérdidas de los datos de entrada

Se inició a trabajar con prioridad puesto que teníamos switches que nos servían para poder generar alguna condición ya definida y dependiendo de estas entradas se decidía si el microcontrolador iba a realizar alguna tarea o si simplemente se reiniciaba o pausaba el programa.

En todos los programas se configuró la pila para poder evitar cualquier error de direccionamiento.

Se pudieron crear nuevas subrutinas como los retardos que nos permitían obtener una frecuencia definida y aproximada a la que se requiere.

Se pudo utilizar un puerto tanto en configuración de entrada como en configuración de salida, además se trabajó con más instrucciones que hasta este momento no se habían utilizado y se aprendió a programar cada uno de los 3 ejercicios con lo que circuitos similares pueden ser programados on el microcontrolador.

8.- REFERENCIAS DE INTERNET

[1].- https://es.wikipedia.org/wiki/Frecuencia_de_reloj

[2].- http://www.alciro.org/alciro/microcontroladores-8051_24/stack-pila_358.htm