



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**

## **Práctica No. 6**

### **Convertidor Analógico/Digital**

**LABORATORIO  
MICROCOMPUTADORAS**

**M.I. RUBÉN ANAYA GARCÍA**

**GRUPO LAB. 08  
GRUPO TEORÍA 05**

## **A L U M N O S**

- FLORES PÉREZ MILNER USHUAÍA  
316137645**
- GUZMÁN RAMÍREZ ALDO Yael  
419049915**
- ROMERO RIVERA GEOVANNI  
316215817**



**Fecha de realización: 13 de abril de 2023, CDMX.  
Fecha de entrega: 11 de mayo de 2023, CDMX.**

**Desarrollo:**

**Realizar los programas solicitados y comprobar su funcionamiento**

- **Ejercicio 1:** Empleando el canal de su elección del convertidor A/D, realizar un programa en el cual, de acuerdo con una entrada analógica que se ingrese por este canal, se represente el resultado de la conversión en un puerto paralelo. Utilizar el arreglo de leds para ver la salida como se muestra en la figura.

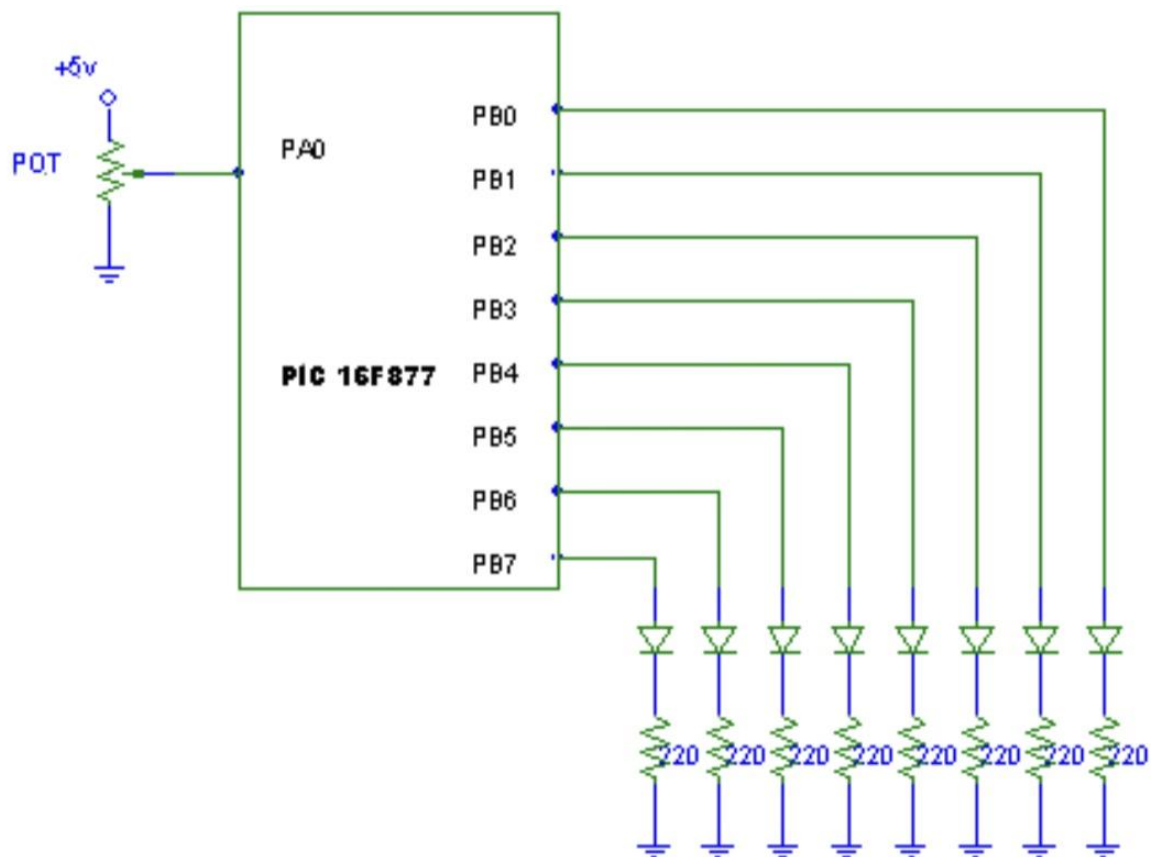


Figura 6.1 Circuito con lectura de una señal analógica

```
E:\Escuela\Labo Micros\Practica 6\Ejercicio 1\Ejercicio 1.asm*
PROCESSOR 16F877          ; Indica la version del procesador
INCLUDE <P16F877.INC>    ; Incluye la libreria de la version del procesador

VAL EQU H'20'            ; Asigna la localidad 20 a VAL

ORG 0                    ; Especifica un origen (Vector de reset)
GOTO INICIO              ;Codigo del programa

ORG 5                    ; Indica origen para inicio del programa
INICIO: CLRWF PORTA      ; Limpia el puerto A
        BSF STATUS, RP0  ; Coloca la bandera RP0 en 1
        BCF STATUS, RP1  ; Coloca la bandera RP1 en 0, asi se mueve al banco 1
        MOVLW H'00'      ; Carga el valor 0 en W
        MOVWF ADCON1     ; Carga los puertos A y E en ADCON1
        CLRF PORTD       ; Limpia el puerto D
        BCF STATUS, RP0  ; Cambia al banco 0
        MOVLW B'11101001' ; Configuracion para ADCON0
        MOVWF ADCON0     ; ADCS1 = 1 ADCS0 = 1 CHS2 = 0 CHS1 = 0 CHS0 = 0
                          ; GO/DONE = 0 - ADON = 1
CICLO:  BSF ADCON0,2      ; Asigna el valor 1 al bit 2 de ADCON0
        CALL RETARDO     ; Llamada a la subrutina RETARDO

ESPERA: BTFSC ADCON0,2    ; Pregunta el si bit 2 de ADCON0 es 0
        GOTO ESPERA      ; NO, Vuelve a ESPERA
        MOVF ADRESH,W    ; SI, Mueve el contenido de ADRESH a W
        MOVWF PORTD      ; Mueve el contenido de W a PORTB
        GOTO CICLO       ; Salta a CICLO

RETARDO:MOVLW H'30'      ; Carga el valor 30 en W
        MOVWF VAL        ; Mueve el contenido de W a VAL

LOOP:   DECFSZ VAL        ; Decrementa el valor de VAL hasta 0
        GOTO LOOP        ; VAL = 0 NO, Salta a LOOP
        RETURN           ; VAL = 0 SI, Vuelve a la subrutina
END                    ; Directiva de fin de programa
```

```
Output
Build Version Control Find in Files
Warning[207] D:\MANY\TAREAS\DECIMO SEMESTRE PT.1\LABO MICROCOMPUTADORAS
Message[302] D:\MANY\TAREAS\DECIMO SEMESTRE PT.1\LABO MICROCOMPUTADORAS
Message[305] D:\MANY\TAREAS\DECIMO SEMESTRE PT.1\LABO MICROCOMPUTADORAS
Executing: "D:\Programas\Microchip\MPASM Suite\mplink.exe" /p16F877 "
MPLINK 4.49, Linker
Device Database Version 1.14
Copyright (c) 1998-2011 Microchip Technology Inc.
Errors      : 0

Loaded D:\Many\Tareas\Decimo semestre Pt.1\Labo Microcomputadoras

Release build of project 'D:\Many\Tareas\Decimo semestre Pt.1\Labo Microcomputadoras'
Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49
Mon May 08 22:52:00 2023

BUILD SUCCEEDED
```

En este programa lo que se realizó fue asignar como entrada uno de los potenciómetros ubicados en el puerto D de la tarjeta, asimismo, marcamos como salida los leds para así poder ver que la conversión se hiciera con éxito, este es un programa introductorio al manejo del convertidor analogico/digital, ya que nos sirve de base para las otras dos actividades que se desarrollaron.

- **Ejercicio 2:** Utilizando el circuito anterior, realizar un programa que indique el rango en el cual se encuentra el voltaje a la entrada del convertidor canal seleccionado. Mostrar el valor en un display de 7 segmentos.

Entrada Analógica Ve	Salida
0 – 0.99 V	0
1.0 – 1.99 V	1
2.0 – 2.99 V	2
3.0 – 3.99 V	3
4.00 – 4.80 V	4
4.80 – 5.00 V	5

**Tabla 6.1**  
Donde  $V_{cc} = 5$  volts

```

E:\Escuela\Labo Micros\Practica 6\Ejercicio 2\Ejercicio 2.asm*
PROCESSOR 16F877      ; Indica la version del procesador
INCLUDE <P16F877.INC> ; Incluye la libreria de la version del procesador

VAL EQU H'20'         ; Asigna la localidad 20 a VAL

ORG 0                 ; Especifica un origen (Vector de reset)
GOTO INICIO           ;Codigo del programa

ORG 5                 ; Indica origen para inicio del programa
INICIO: CLRF PORTA     ; Limpia el puerto A
        BSF STATUS, RP0 ; Coloca la bandera RP0 en 1
        BCF STATUS, RP1 ; Coloca la bandera RP1 en 0, asi se mueve al banco 1
        MOVLW H'00'     ; Carga el valor 0 en W
        MOVWF ADCON1    ; Carga los puertos A y E en ADCON1
        CLRF PORTD      ; Limpia el puerto D
        BCF STATUS, RP0 ; Cambia al banco 0
        MOVLW B'11101001' ; Configuracion para ADCON0
        MOVWF ADCON0    ; ADCS1 = 1 ADCS0 = 1 CHS2 = 0 CHS1 = 0 CHS0 = 0
                        ; GO/DONE = 0 - ADON = 1
CICLO:   BSF ADCON0,2   ; Asigna el valor 1 al bit 2 de ADCON0
        CALL RETARDO   ; Llamada a la subrutina RETARDO

ESPERA:  BTFSC ADCON0,2 ; Pregunta si el bit 2 de ADCON0 es 0
        GOTO ESPERA    ; NO, Vuelve a ESPERA
        MOVF ADRESH,0  ; SI, Mueve el contenido de ADRESH a W (VE = VOLTAJE DE ENTRADA)
        SUBLW 130H     ; Resta 130 - W (0/5 VCC = 0CCH)
        BTFSC STATUS,0 ; Pregunta si el bit 0 de STATUS es 0
        GOTO SALIDA0   ; NO, Salta a SALIDA0 (VE < 1/5 VCC)
        MOVF ADRESH,0  ; SI, Mueve el contenido de ADRESH a W

```

```
E:\Escuela\Labo Micros\Practica 6\Ejercicio 2\Ejercicio 2.asm*
; NO, Salta a SALIDA0 (VE < 1/5 VCC)
GOTO SALIDA0
; SI, Mueve el contenido de ADRESH a W
MOVE ADRESH,0
; Resta 265 - W (1/5 VCC = 198H)
SUBLW 265H
; Pregunta si el bit 0 de STATUS es 0
BTFSC STATUS,0
; NO, Salta a SALIDA1 (1/5 VCC < VE < 2/5 VCC)
GOTO SALIDA1
; SI, Mueve el contenido de ADRESH a W
MOVE ADRESH,0
; Resta 398 - W (2/5 VCC = 265H)
SUBLW 398H
; Pregunta si el bit 0 de STATUS es 0
BTFSC STATUS,0
; NO, Salta a SALIDA2 (2/5 VCC < VE < 3/5 VCC)
GOTO SALIDA2
; SI, Mueve el contenido de ADRESH a W
MOVE ADRESH,0
; Resta 3CAH - W (3/5 VCC = 332H)
SUBLW 3CAH
; Pregunta si el bit 0 de STATUS es 0
BTFSC STATUS,0
; NO, Salta a SALIDA3 (3/5 VCC < VE < 4/5 VCC)
GOTO SALIDA3
; SI, Mueve el contenido de ADRESH a W
MOVE ADRESH,0
; Resta 3F7H - W (VCC = 3FFH)
SUBLW H'3F7'
; Pregunta si el bit 0 de STATUS es 0
BTFSC STATUS,0
; NO, Salta a SALIDA4 (4/5 VCC < VE < 5/5 VCC)
GOTO SALIDA4
; SI, Mueve el contenido de ADRESH a W
MOVE ADRESH,0
; Resta 3FFH - W
SUBLW H'3FF'
; Pregunta si el bit 0 de STATUS es 0
BTFSC STATUS,0
; NO, Salta a SALIDA5
GOTO SALIDA5

SALIDA0:    MOVLW H'0'    ; Mueve 0 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

SALIDA1:    MOVLW H'1'    ; Mueve 1 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO
```

```
E:\Escuela\Labo Micros\Practica 6\Ejercicio 2\Ejercicio 2.asm*
; Salta a CICLO
GOTO CICLO

SALIDA1:    MOVLW H'1'    ; Mueve 1 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

SALIDA2:    MOVLW H'2'    ; Mueve 2 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

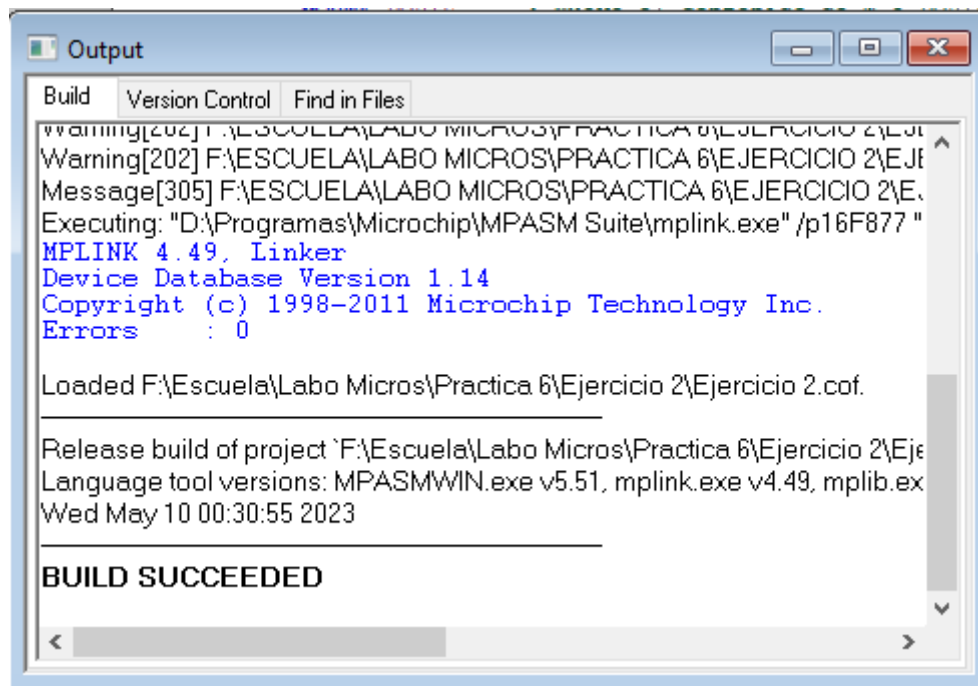
SALIDA3:    MOVLW H'3'    ; Mueve 3 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

SALIDA4:    MOVLW H'4'    ; Mueve 4 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

SALIDA5:    MOVLW H'5'    ; Mueve 5 a W
            MOVWF PORTD  ; Mueve el contenido de W a PORTD
            GOTO CICLO   ; Salta a CICLO

RETARDO:    MOVLW H'30'   ; Carga el valor 30 en W
            MOVWF VAL     ; Mueve el contenido de W a VAL

LOOP:       DECFSZ VAL    ; Decrementa el valor de VAL hasta 0
            GOTO LOOP     ; VAL = 0 NO, Salta a LOOP
            RETURN        ; VAL = 0 SI, Vuelve a la subrutina
END          ; Directiva de fin de programa
```



En este programa, haciendo uso del ejercicio anterior, configurando el puerto D como entrada para usar el potenciómetro, y en este caso para poder ver la salida en los display de siete segmentos de manera correcta lo que hacemos es que durante la conversión también realizamos una comparación entre los valores que suministramos para así pasarlos a subrutinas que dependiendo de dicha entrada, puedan mostrar un numero del 1 al 5, así, sabemos cuanta cantidad de voltaje estamos enviando a nuestra tarjeta, ya no solo por intuición u observación en la barra de leds.

- **Ejercicio 3:** Realizar un programa, de manera que identifique cuál de tres señales analógicas que ingresan al convertidor A/D es mayor que las otras dos; representar el resultado de acuerdo con el contenido de la tabla.

Señal	PB2	PB1	PB0
Ve1>Ve2 y Ve3	0	0	1
Ve2>Ve1 y Ve3	0	1	1
Ve3>Ve1 y Ve2	1	1	1

Tabla 6.2

```
F:\Escuela\Labo Micros\Practica 6\Ejercicio 3.asm*
PROCESSOR 16F877          ; Indica la version del procesador
INCLUDE <P16F877.INC>    ; Incluye la libreria de la version del procesador

valor1 equ H'21'          ;Asigna el valor de 21 a valor1
valor2 equ H'22'          ;Asigna el valor de 22 a valor2
valor3 equ H'23'          ;Asigna el valor de 23 a valor3

cte1 equ 1H               ; Asigna el valor de 1 a cte1
cte2 equ 2H               ; Asigna el valor de 2 a cte2
cte3 equ 3H               ; Asigna el valor de 3 a cte3

vel equ H'31'             ; Asigna el valor de 31 a vel
ve2 equ H'35'             ; Asigna el valor de 35 a ve2
ve3 equ H'36'             ; Asigna el valor de 36 a ve3
mayor equ H'37'           ; Asigna el valor de 37 a mayor

ORG 0                     ;Especifica un origen (vector de reset)
GOTO INICIO               ;Código del programa

ORG 5                     ;Indica origen para inicio del programa
INICIO: CLRF PORTA         ; Limpia el puerto A
        CLRF PORTD        ; Limpia el puerto D
        CLRF PORTE        ; Limpia el puerto E
        BSF STATUS, RP0   ; Coloca la bandera RP0 en 1
        BCF STATUS, RP1   ; Coloca la bandera RP1 en 0, así se mueve al banco 1
        MOVLW H'00'       ; Carga el valor 0 en W
        MOVWF TRISD       ; Configura puerto D como salida
        MOVLW H'00'       ; Carga el valor 0 en W
        MOVWF ADCON1      ; Carga los puertos A y E en ADCON1
        BCF STATUS, RP0   ; Cambia al banco 0

E1:     MOVLW 0x69         ; Carga el valor 69 en W
        MOVWF ADCON0      ; Mueve el contenido de W a ADCON0
        BSF ADCON0, 2     ; Establece en 1 el bit 2 de ADCON0
        CALL ESPERA       ; Llamada a la subrutina ESPERA
        MOVF ADRESH, W    ; Mueve el contenido de ADRESH a W
        MOVWF vel         ; Mueve el contenido de W a vel
```

```
F:\Escuela\Labo Micros\Practica 6\Ejercicio 3.asm*
E2:    MOVLW 0x71    ; Carga el valor 71 en W
        MOVWF ADCON0 ; Mueve el contenido de W a ADCON0
        BSF ADCON0,2 ; Establece en 1 el bit 2 de ADCON0
        CALL ESPERA  ; Llamada a la subrutina ESPERA
        MOVF ADRESH,W ; Mueve el contenido de ADRESH a W
        MOVWF ve2    ; Mueve el contenido de W a ve2

E3:    MOVLW 0x79    ; Carga el valor 79 en W
        MOVWF ADCON0 ; Mueve el contenido de W a ADCON0
        BSF ADCON0,2 ; Establece en 1 el bit 2 de ADCON0
        CALL ESPERA  ; Llamada a la subrutina ESPERA
        MOVF ADRESH,W ; Mueve el contenido de ADRESH a W
        MOVWF ve3    ; Mueve el contenido de W a ve3

Comp:  MOVE ve1,W    ; Mueve el contenido de ve1 a W
        SUBWF ve2,W  ; Resta ve2 - W
        BTFSS STATUS,C ; Pregunta si el bit C de STATUS es 1
        GOTC MAYOR_1 ; NO, Salta a MAYOR_1
        GOTC MAYOR_2 ; SI, Salta a MAYOR_2

MAYOR_1: MOVE ve1,W    ; Mueve el contenido de ve1 a W
        MOVWF mayor    ; Mueve el contenido de W a mayor
        SUBWF ve3,W    ; Resta ve3 - W
        BTFSS STATUS,C ; Pregunta si el bit C de STATUS es 1
        GOTC ASIG1     ; NO, Salta a ASIG1
        MOVF ve3,W     ; SI, Mueve el contenido de ve3 a W
        MOVWF mayor    ; Mueve el contenido de W a mayor
        GOTC ASIG3     ; Salta a ASIG3

MAYOR_2: MOVE ve2,W    ; Mueve el contenido de ve2 a W
        MOVWF mayor    ; Mueve el contenido de W a mayor
        SUBWF ve3,W    ; Resta ve3 - W
        BTFSS STATUS,C ; Pregunta si el bit C de STATUS es 1
        GOTC ASIG2     ; NO, Salta a ASIG2
        MOVF ve3,W     ; SI, Mueve el contenido de ve3 a W
        MOVWF mayor    ; Mueve el contenido de W a mayor
        GOTC ASIG3     ; Salta a ASIG3
```



```
F:\Escuela\Labo Micros\Practica 6\Ejercicio 3.asm*
MOVWF mayor      ; Mueve el contenido de W a mayor
GOTO ASIG3        ; Salta a ASIG3

ASIG1: MOVWL 0x01   ; Mueve 01 a W
      MOVWF PORTD  ; Mueve el contenido de W a PORTD
      GOTO E1      ; Salta a E1

ASIG2: MOVWL 0x03   ; Mueve 03 a W
      MOVWF PORTD  ; Mueve el contenido de W a PORTD
      GOTO E1      ; Salta a E1

ASIG3: MOVWL 0x07   ; Mueve 07 a W
      MOVWF PORTD  ; Mueve el contenido de W a PORTD
      GOTO E1      ; Salta a E1

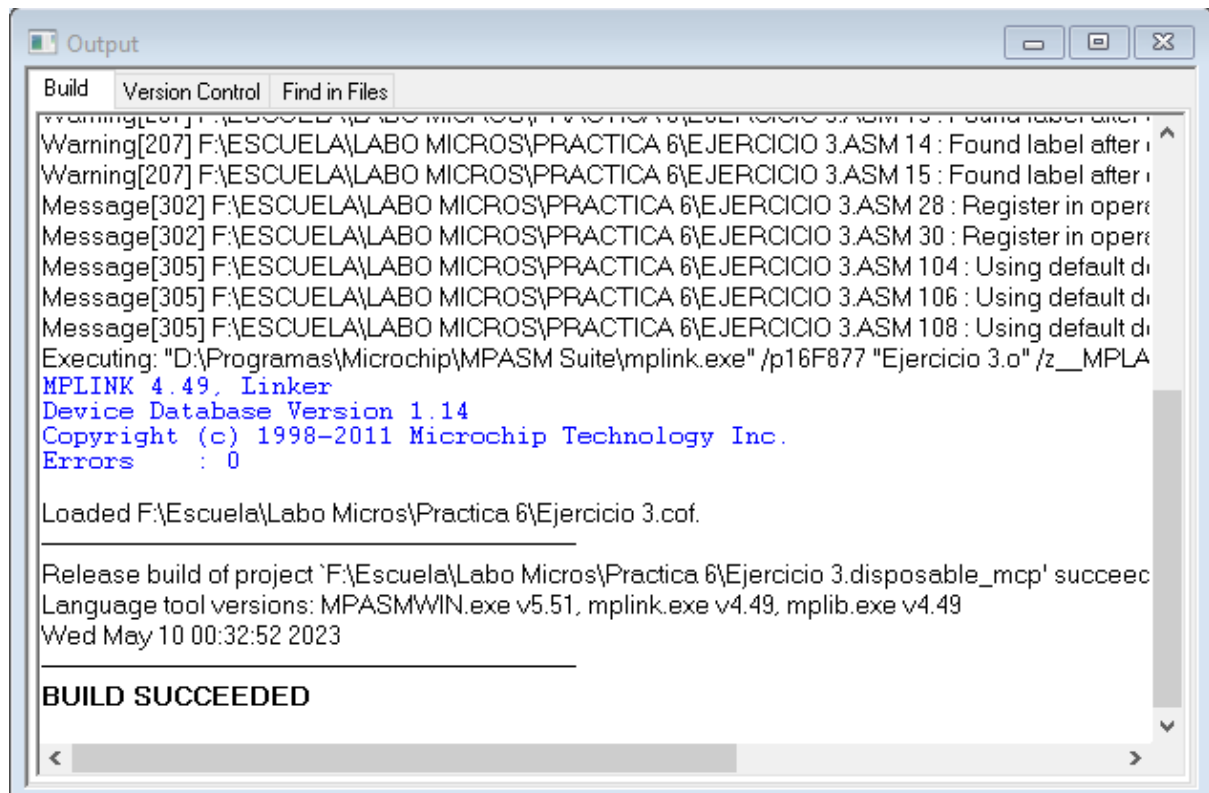
ESPERA: CALL RETARDO ; Llamada a la subrutina RETARDO
      BTFSC ADCON0,2 ; Pregunta si el bit 2 de ADCON0 es 0
      GOTO ESPERA   ; NO, Salta a ESPERA
      RETURN        ; SI, Regresa a la subrutina

RETARDO: MOVWL cte1  ; Guarda en el registro W el valor de cte1
      MOVWF valor1  ; Mueve el contenido del registro W al registro valor1

TRES:   MOVWL cte2   ; Guarda en el registro W el valor de cte2
      MOVWF valor2  ; Mueve el contenido del registro W al registro valor2

DOS:    MOVWL cte3   ; Guarda en el registro W el valor de cte3
      MOVWF valor3  ; Mueve el contenido del registro W al registro valor3

UNO:    DECFSZ valor3 ; Decrementa el registro valor3 hasta cero
      GOTO UNO      ; Salta a UNO
      DECFSZ valor2  ; Decrementa el registro valor2 hasta cero
      GOTO DOS      ; Salta a DOS
      DECFSZ valor1  ; Decrementa el registro valor1 hasta cero
      GOTO TRES      ; Salta a TRES
      RETURN        ; Regresa de la subrutina
      END           ; Directiva de fin de programa
```



```
Output
Build Version Control Find in Files
Warning[207] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 14 : Found label after :
Warning[207] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 15 : Found label after :
Message[302] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 28 : Register in oper
Message[302] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 30 : Register in oper
Message[305] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 104 : Using default di
Message[305] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 106 : Using default di
Message[305] F:\ESCUELA\LABO MICROS\PRACTICA 6\EJERCICIO 3.ASM 108 : Using default di
Executing: "D:\Programas\Microchip\MPASM Suite\mplink.exe" /p16F877 "Ejercicio 3.o" /z__MPLA
MPLINK 4.49, Linker
Device Database Version 1.14
Copyright (c) 1998-2011 Microchip Technology Inc.
Errors : 0

Loaded F:\Escuela\Labo Micros\Practica 6\Ejercicio 3.cof.

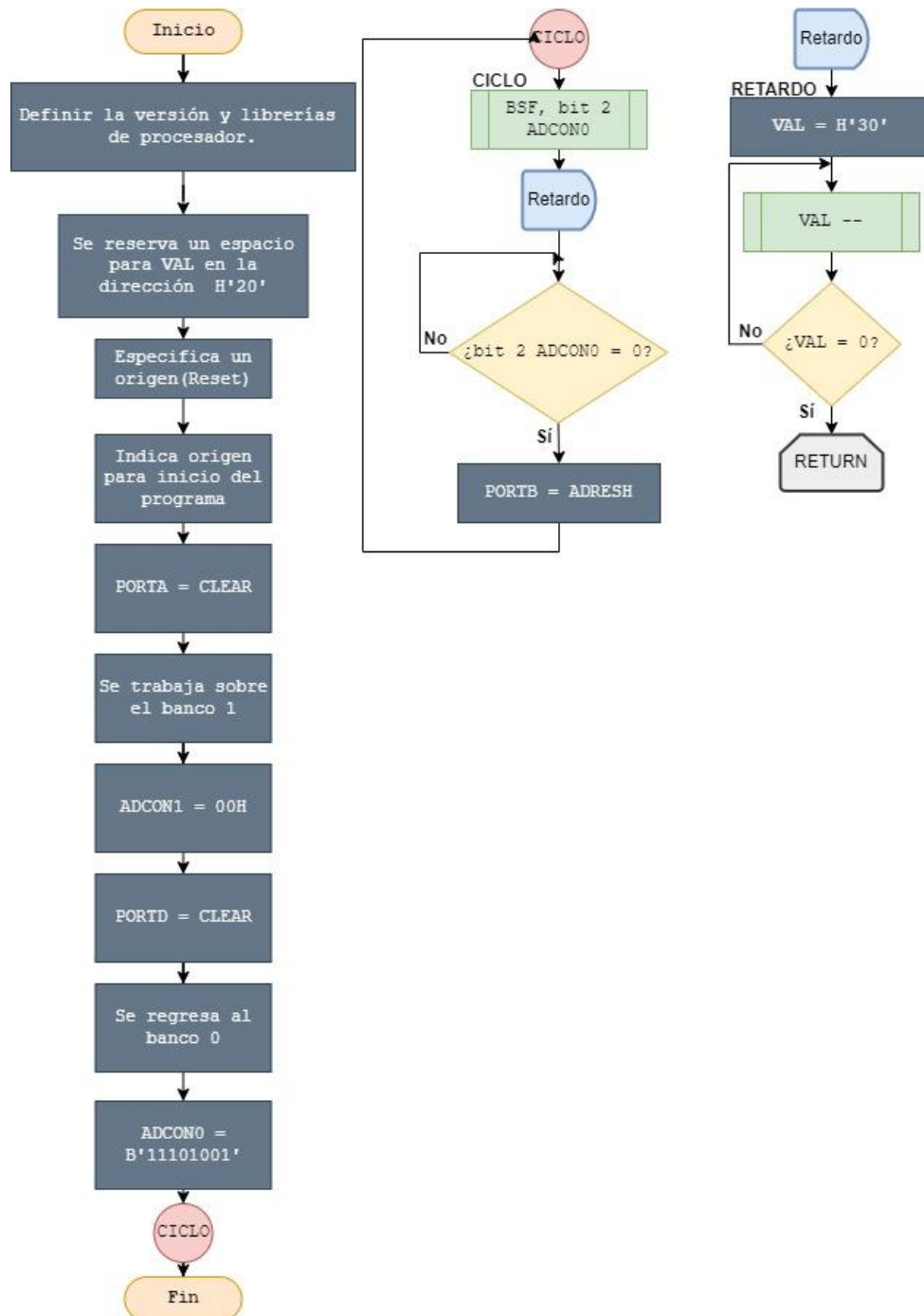
Release build of project 'F:\Escuela\Labo Micros\Practica 6\Ejercicio 3.disposable_mcp' succeec
Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49
Wed May 10 00:32:52 2023

BUILD SUCCEEDED
```

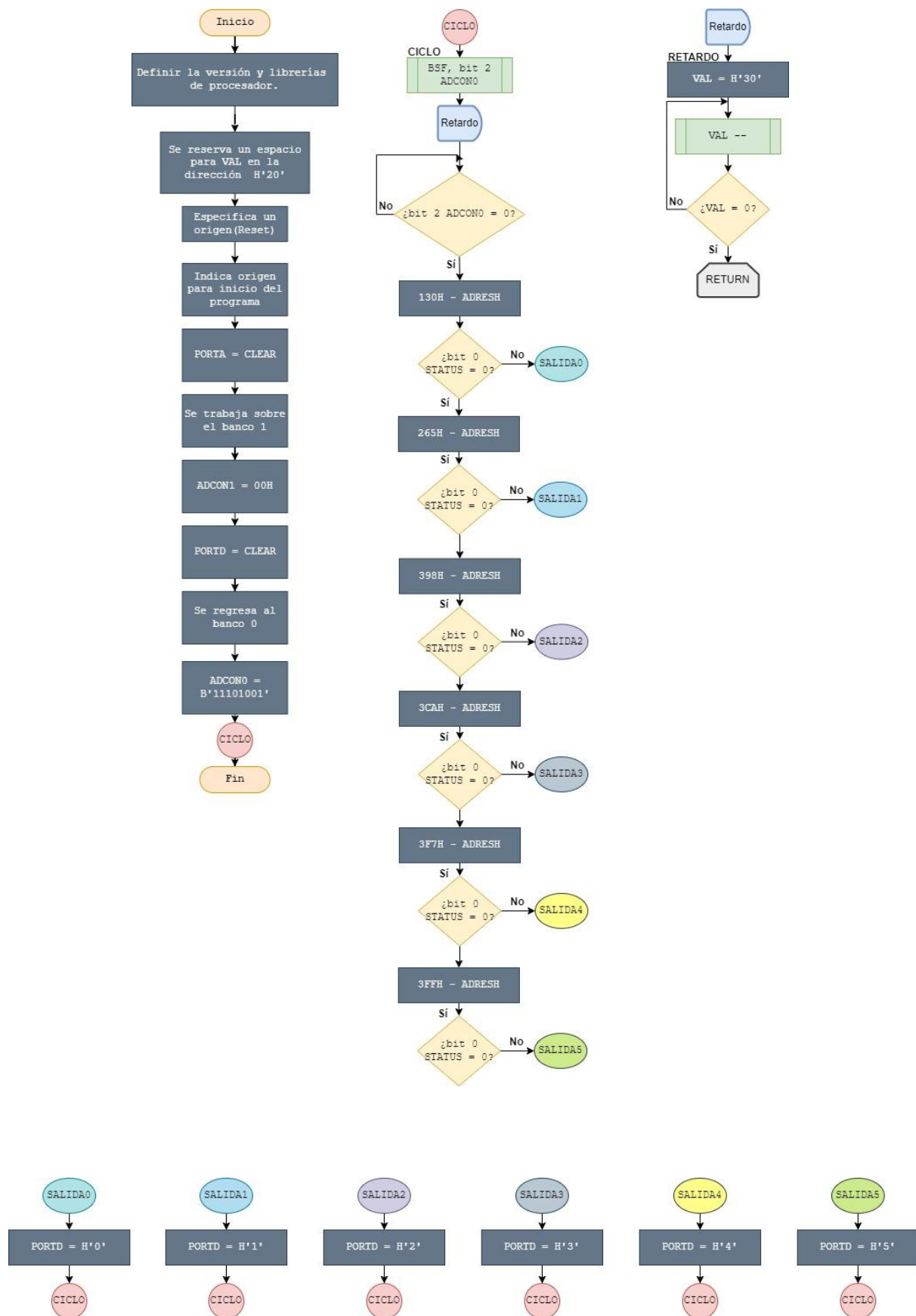
Finalmente, para este programa se implementó un algoritmo de comparación de tres valores como otro que se ha hecho en prácticas anteriores, con ello, solo tenemos que comparar la entrada que estamos recibiendo en los puertos A y E, es decir, en los tres potenciómetros para, después pasar ese valor recibido a nuestro comparador, establecer cuál de ellos es el mayor mediante las rutinas MAYOR\_1 y MAYOR\_2, con las que sí ni el primero valor ni el segundo es el mayor, simplemente decimos que el tercero es el mayor. Así, determinado nuestro valor, lo pasamos a la salida en el puerto D como código binario a los leds y como decimal al display de 7 segmentos representado de la misma forma (1 para el primer valor, 3 para el segundo y 7 para el tercero)

## Algoritmos

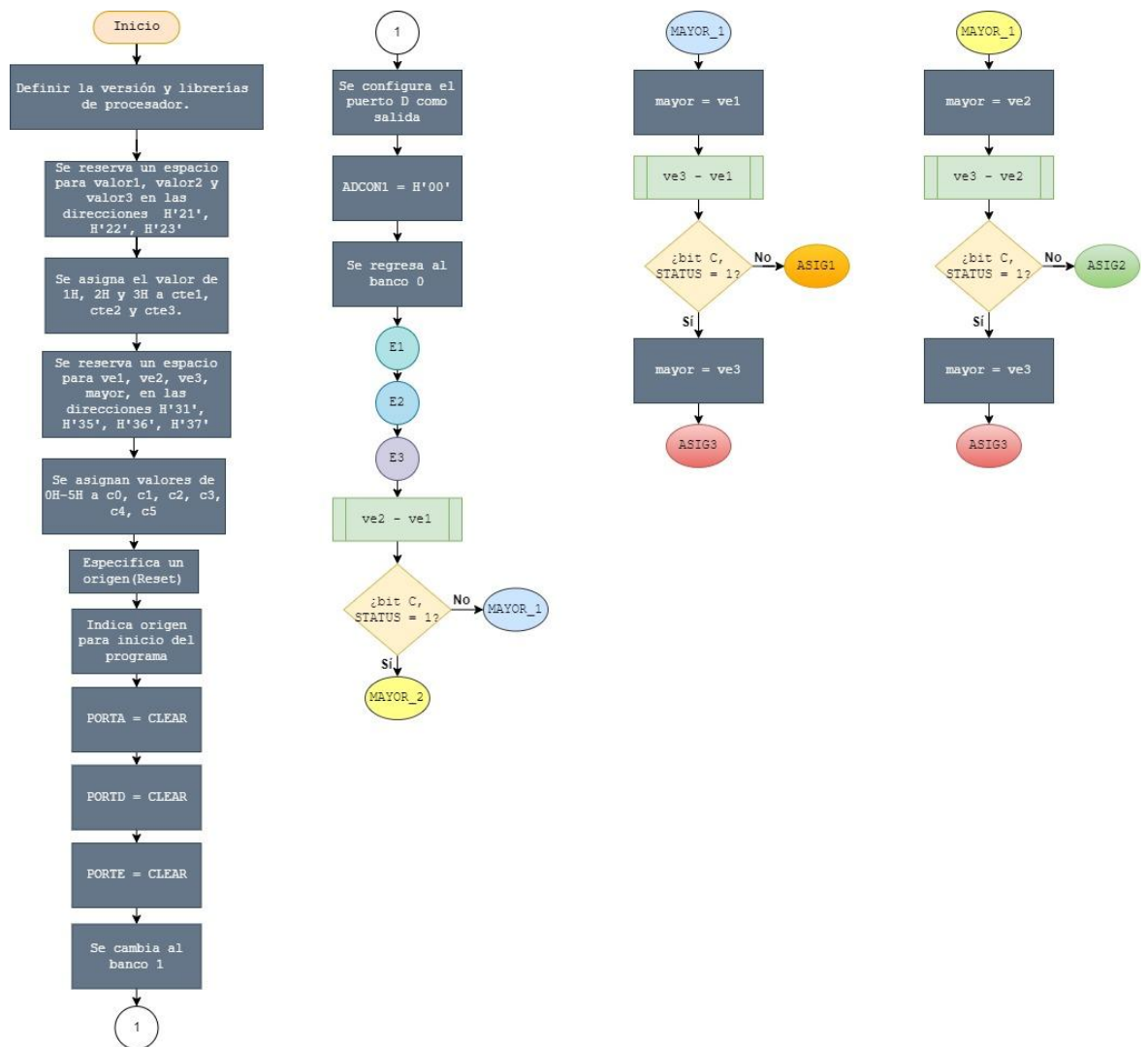
### - Ejercicio 1

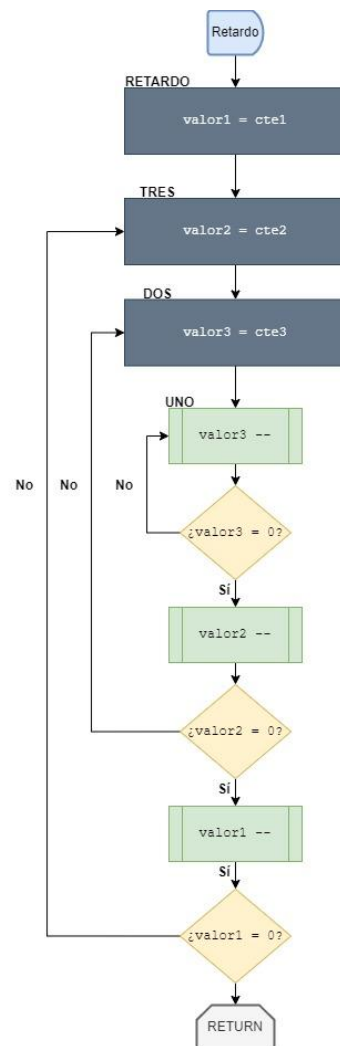
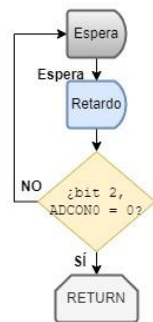
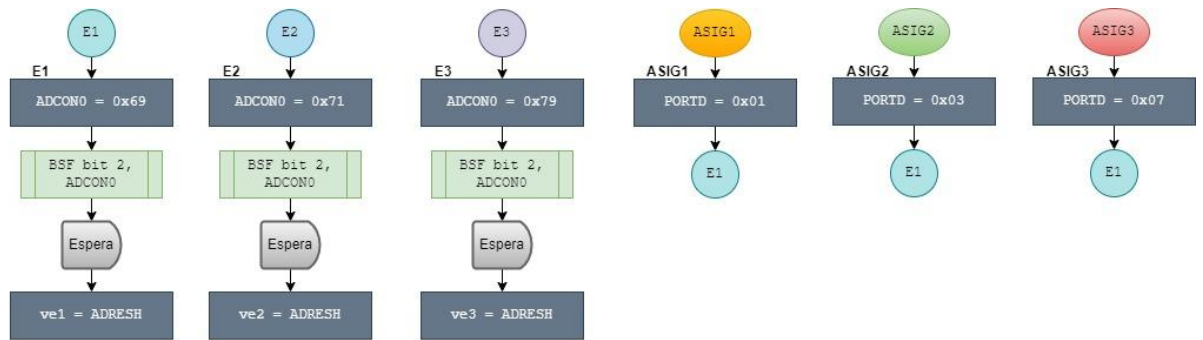


## - Ejercicio 2



### - Ejercicio 3





## **Conclusiones:**

**Flores Perez Milner Ushuaía:** En esta práctica se consolidaron los conocimientos aprendidos en las sesiones anteriores, pues también se hizo uso del manejo de puertos, así como la configuración de los mismos, pero ahora con enfoque en el uso del convertidor analógico/digital, que nos permite tomar como entrada un impulso eléctrico del exterior y como salida hacer uso de los componentes en la tarjeta, como los leds. Con ayuda del ejemplo y los programas realizados en prácticas anteriores, fue posible crear diferentes programas que recibieran un voltaje de entrada y manipularan la información obtenida, haciendo comparaciones y, de esta forma, obtener salidas que cumplieran nuestros requerimientos.

**Guzman Ramirez Aldo Yael:** Seguimos con el aprendizaje de todo lo que puede realizar nuestro microcontrolador, esta vez nos tocó experimentar cómo recibe el voltaje y transformar esta energía de una señal analógica a digital. Por otra parte también experimentamos con la intensidad con la que esta se transmite a través de los leds. Por último analizamos tres señales y con ayuda igual de nuestro leds representamos cual de estas era más grande comparado cada una. En un principio pensé que nuestro microcontrolador nos dispensaba el voltaje de la misma manera pero una vez que le preguntamos al maestro esta teoría fue descartada por que quizá solo estaba un poco descalibrado el microcontrolador.

**Romero Rivera Geovanni:** Para esta práctica, comprender el primer ejercicio fue vital ya que de ahí nos basamos para poder desarrollar los otros dos. Al principio fue muy difícil comprender cómo funciona el convertidor analógico/digital pero más que nada entender como programarlo, sin embargo, gracias a la ayuda del profesor con el primer ejercicio resultó más fácil ver cómo se comporta el manejo de puertos y los bits más/menos significativos a la hora de configurar los puertos de entrada y salida. Con ellos crear el segundo ejercicio fue solo cuestión de establecer comparadores para verificar si nuestra entrada era igual o menor a la que esperábamos y por último, en el tercer ejercicio solo se implementó un comparador de tres valores. Con el avance de las prácticas se ha notado el aprendizaje al momento de emplear códigos o algoritmos usados en anteriores ejercicios.