Instituto Politécnico Nacional

Escuela Superior de Cómputo

Alumnos:

Esquivel Pérez Jonathan Alfredo
Hernández López Ángel Zait
Salgado Gallegos Jesús
Sánchez Pizano Irving Daniel

Profesor:

Pérez Pérez José Juan

Unidad de aprendizaje:

Introducción a los microcontroladores

Práctica 7

Convertidor ADC

Grupo: 3CM8

Objetivo

Diseñar un circuito ADC (Analógico a Digital) en en Proteus IDE capaz de recibir como entrada un número desde un teclado el cual se Implementar un script en lenguaje ensamblador que permita captar un número del teclado y convertirlo, por medio de un ADC , a un display, el cual se encargará de mostrar la información recibida.

Introducción

ADC

Un conversor o convertidor de señal analógica a digital (Conversor Analógico Digital, CAD; Analog-to-Digital Converter, ADC) es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular.

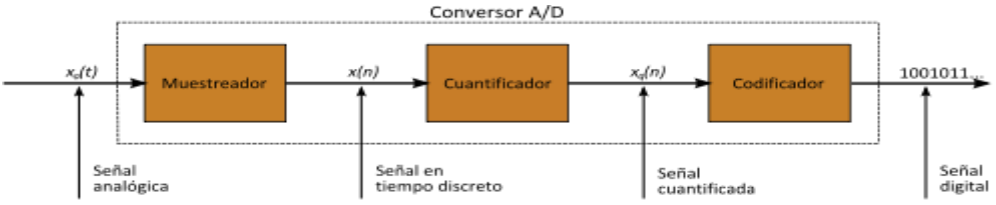

Imagen 1. Convertidor analógico digital.

Interrupciones

Una interrupción es un evento asíncrono que provoca detener la ejecución del código actual en base a una prioridad (cuanto más importante es la interrupción, mayor será su prioridad, lo que hará que una interrupción de menor prioridad se suspenda). La rutina que se ejecuta durante una interrupción se denomina “Rutina de servicio de interrupción” o “Interrupt Service Routine” por sus siglas en inglés ISR. Cuando ésta se produce, el microcontrolador va a la dirección 04h de programa y ejecuta lo que encuentre a partir de allí hasta encontrarse con la instrucción RETFIE que le hará abandonar la interrupción.

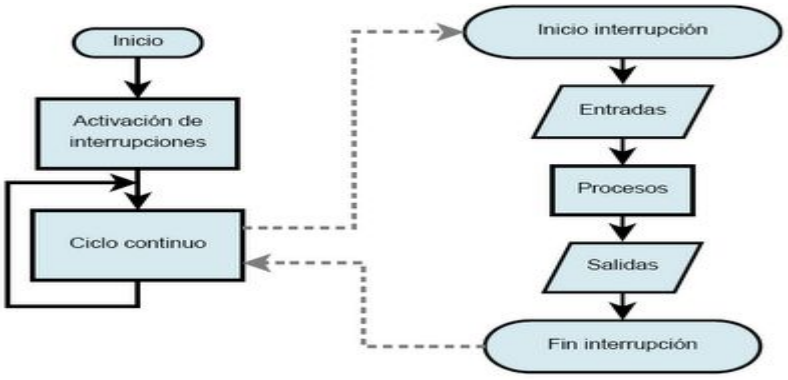

Imagen 2. Diagrama de flujo de interrupciones.

Material y equipo utilizado

- Software IDE Proteus
- Resistencias
- Teclado digital
- Display de 8 segmentos
- Voltímetro
- Cable
- Resistencias
- Barra de LED's
- Microcontrolador ATMEGA8535
- Software AVR Studio

Que se realizará en la práctica

Se implementará un ADC (Analógico a Digital) el cuál recibirá un dato que se mostrará posteriormente en un display dependiendo qué número haya elegido el usuario. Así mismo usando un voltímetro se sabrá qué tanto voltaje llega hacia el ATMega. Una vez ingresada la combinación se verá reflejado en el display donde se podrá apreciar el número en su formato decimal.

Desarrollo de la práctica, incluyendo diagramas de flujo y codificación comentada

Código

```
1.  .include"m8535def.inc"
2.  .def adl = r17
3.  .def adh = r16
4.  .def tecla = r18
5.  rjmp  Start
6.  .ORG $0E
7.  RJMP CONV
8.
9.  Start:
10.     LDI R16,LOW(RAMEND) ;carga bytes bajos de la sram en r16
11.     OUT SPL,R16 ;coloca el r16 en sph, byte bajo de la pila
12.     LDI R16,HIGH(RAMEND) ;carga bytes altos de la sram en r16
13.     OUT SPH,R16 ;coloca el r16 en sph, byte alto de la pila
14.     SER R16
15.     OUT DDRD,R16
16.     OUT DDRB,R16
17.     OUT DDRC,R16
18.     LDI R16,$ED ; 1110 1101
19.     OUT ADCSRA,R16
20.     ldi r16,$20; 0010 0000, con $00 la alineacion es a la izquierda
21.     out ADMUX,r16
22.     clr tecla
23.     SEI
24.
25. Loop:
26.     OUT PORTD,adl
27.     OUT PORTB,adh
```

```
    out portc,tecla
    rjmp  Loop ;Salta a Loop

CONV:
    IN adl,ADCL
    IN adh,ADCH
    cpi adl,$C0
    breq compara00 ;Realiza un desvio a compara00 si los registros son iguales
    cpi adl,$40
    breq compara01 ;Realiza un desvio a compara01 si los registros son iguales
    cpi adl,$80
    breq compara02 ;Realiza un desvio a compara02 si los registros son iguales
    cpi adl,$00
    breq compara03 ;Realiza un desvio a compara03 si los registros son iguales
    brne nada
    RETI

nada:
    clr tecla
    reti

compara00:
    cpi adh,$C8
    breq uno ;Realiza un desvio a uno si los registros son iguales
    cpi adh,$54
    breq mas ;Realiza un desvio a mas si los registros son iguales
    cpi adh,$B7
    breq cuatro ;Realiza un desvio a cuatro si los registros son iguales
    cpi adh,$89
    breq ocho ;Realiza un desvio a ocho si los registros son iguales
    reti

compara01:
    cpi adh,$D8
    breq on_c ;Realiza un desvio a on_c si los registros son iguales
    cpi adh,$52
    breq menos ;Realiza un desvio a menos si los registros son iguales
    cpi adh,$4F
    breq equis ;Realiza un desvio a equis si los registros son iguales
    cpi adh,$6F
    breq nueve ;Realiza un desvio a nueve si los registros son iguales
    reti

compara02:
    cpi adh,$A2
    breq cero ;Realiza un desvio a cero si los registros son iguales
    cpi adh,$99
    breq dos ;Realiza un desvio a dos si los registros son iguales
    cpi adh,$79
    breq tres ;Realiza un desvio a tres si los registros son iguales
    cpi adh,$8F
    breq cinco ;Realiza un desvio a cinco si los registros son iguales
    cpi adh,$AE
    breq siete ;Realiza un desvio a siete si los registros son iguales
```

```
    cpi adh,$4D
    breq div ;Realiza un desvio a div si los registros son iguales
    reti

compara03:
    cpi adh,$7F
    breq igual ;Realiza un desvio a igual si los registros son iguales
    cpi adh,$73
    breq seis ;Realiza un desvio a seis si los registros son iguales
    reti

on_c:
    ldi tecla, $39 ;Guarda la C en el registro 18 para mostrar el resultado en el display
    reti

cero:
    ldi tecla, $3F ;Guarda el cero en el registro 18 para mostrar el resultado en el display
    reti

  uno:
      ldi tecla, $06 ;Guarda el uno en el registro 18 para mostrar el resultado en el display
      reti

  dos:
      ldi tecla, $5B ;Guarda el dos en el registro 18 para mostrar el resultado en el display
      reti

  tres:
      ldi tecla, $4F ;Guarda el tres en el registro 18 para mostrar el resultado en el display
      reti

  cuatro:
      ldi tecla, $66 ;Guarda el cuatro en el registro 18 para mostrar el resultado en el display
      reti

  cinco:
      ldi tecla, $6D ;Guarda el cinco en el registro 18 para mostrar el resultado en el display
      reti

  seis:
      ldi tecla, $7D ;Guarda el seis en el registro 18 para mostrar el resultado en el display
      reti

  siete:
      ldi tecla, $27 ;Guarda el siete en el registro 18 para mostrar el resultado en el display
```

```
127.        reti
128.
129.    ocho:
130.        ldi tecla, $7f ;Guarda el ocho en el registro 18 para mostrar el
    resultado en el display
131.        reti
132.
133.    nueve:
134.        ldi tecla, $6f ;Guarda el nueve en el registro 18 para mostrar el
    resultado en el display
135.        reti
136.
137.    mas:
138.        ldi tecla, $09 ;Guarda el mas en el registro 18 para mostrar el
    resultado en el display
139.        reti
140.
141.    menos:
142.        ldi tecla, $40 ;Guarda el menos en el registro 18 para mostrar el
    resultado en el display
143.        reti
144.
145.    equis:
146.        ldi tecla, $76 ;Guarda la X en el registro 18 para mostrar el resultado
    en el display
147.        reti
148.
149.    div:
150.        ldi tecla, $52 ;Guarda el / en el registro 18 para mostrar el resultado
    en el display
151.        reti
152.
153.    igual:
154.        ldi tecla,$48 ;Guarda el = en el registro 18 para mostrar el resultado
    en el display
155.        reti
```

Representación de números

Lógica de los número para mostrarlos en el display de 7 segmentos

```
numBin = numHex
Representa el valor de ADCL
    simbolo - numBin = numHex -> numHex
    Representa el simbolo que queremos ver, seguido del valor que tiene al
presionar la tecla
    en el registro correspondiente a ADCH, seguido de su numero hexadecimal
equivalente
    y por último el valor hexadecimal del símbolo para el display de 7
segmentos

11000000 = $C0
    1 - 11001000 = $C8 -> $06
```

```
      + - 01010100 = $54 -> $09
      4 - 10110111 = $B7 -> $66
      8 - 10001001 = $89 -> $7f

01000000 = $40
      C - 11011000 = $D8 -> $39
      - - 01010010 = $52 -> $40
      x - 01001111 = $4F -> $76
      9 - 01101111 = $6F -> $6F

10000000 = $80
      0 - 10100010 = $A2 -> $3F
      2 - 10011001 = $99 -> $5B
      3 - 01111001 = $79 -> $4F
      5 - 10001111 = $8F -> $6D
      7 - 10101110 = $AE -> $27
      / - 01001101 = $4D -> $52


00000000 = $00
      = - 01111111 = $7F -> $41
      6 - 01110011 = $73 -> $7D
```

Simulación

Link de la simulación:
https://drive.google.com/file/d/1dl3oHkAew_9EN01NMUCHPhEl4sZG3eMQ/view?usp=sharing

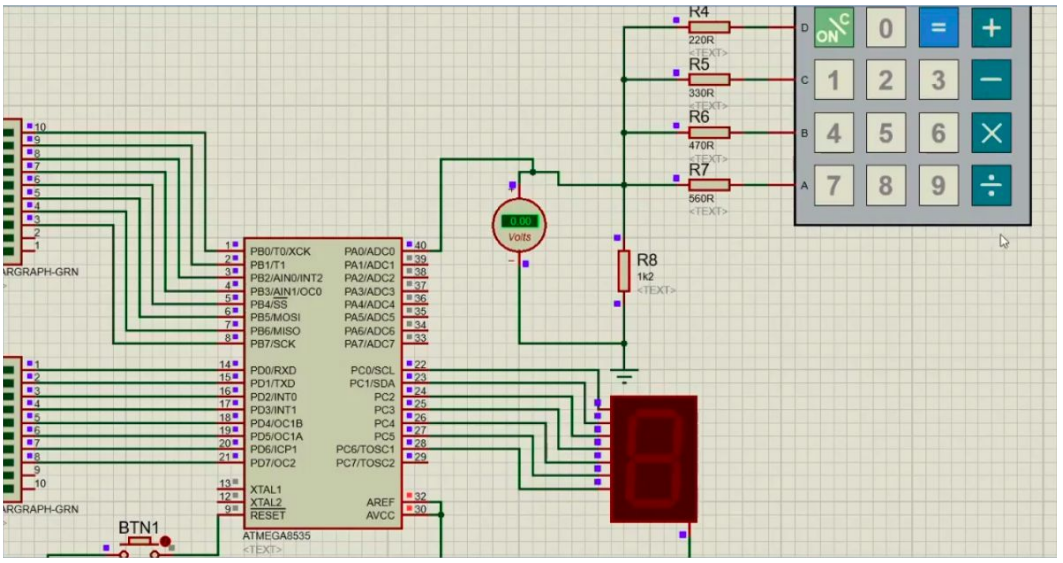

Imagen 3. Convertidor analógico digital apagado

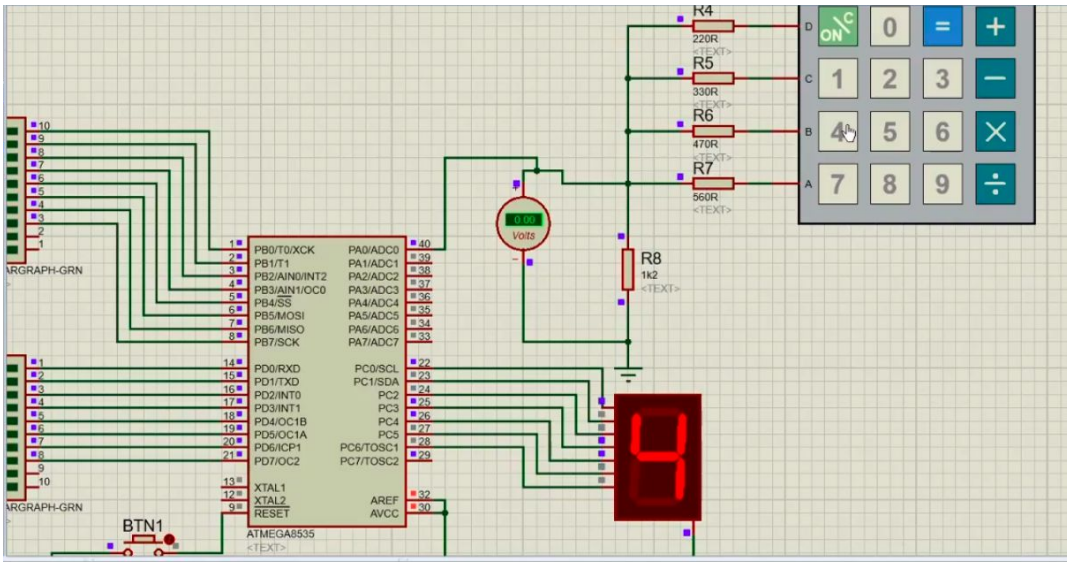

Imagen 4. Convertidor analógico digital funcionando

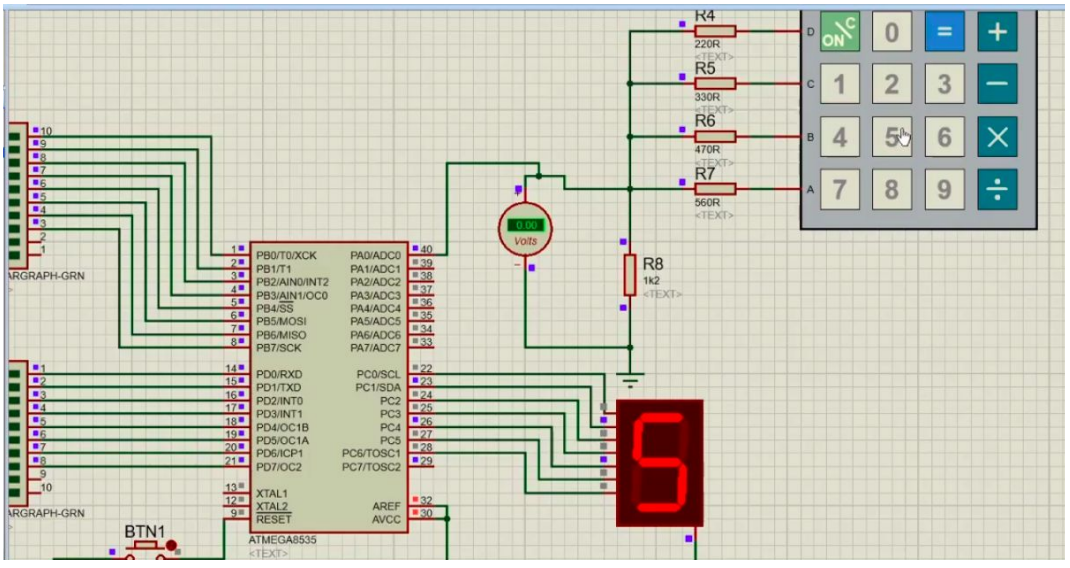

Imagen 5. Convertidor analógico digital funcionando

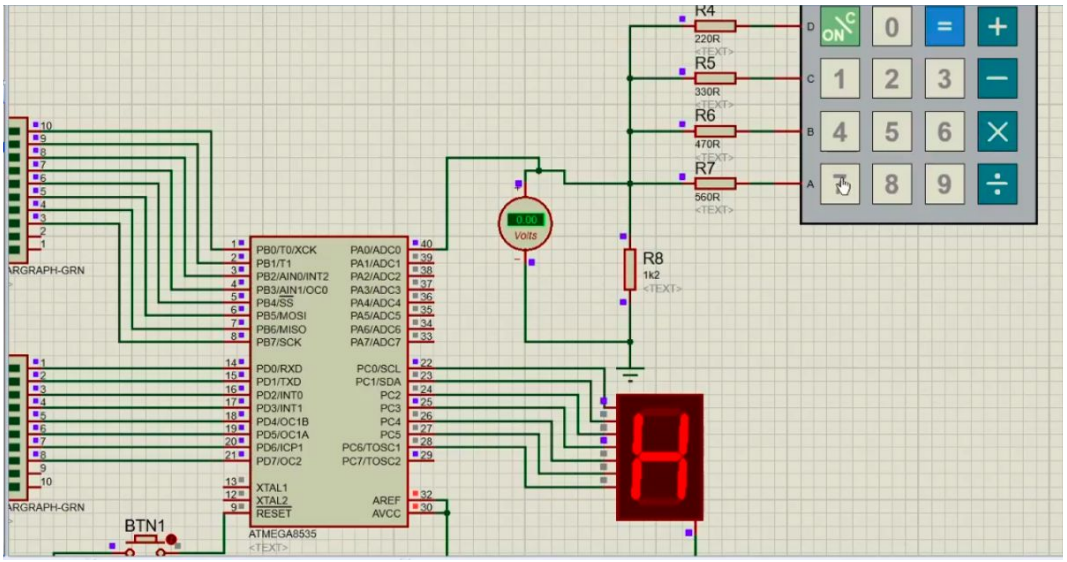

Imagen 6. Convertidor analógico digital funcionando

Conclusiones

Esquivel Pérez Jonathan Alfredo:
En esta práctica se hicieron uso nuevamente de interrupciones en el donde por medio de flancos de subida o bajada, el microcontrolador recibe una interrupción para que cambie su rutina o se establezca un reset a los valores iniciales. La habilitación de las interrupciones se hicieron mediante el uso de bits que detectan la interrupción. Una vez captado el valor en la matriz, el programa detecta el flanco y se habilita cierta sección de código según lo que se haya detectado.

Hernández López Ángel Zait:
En esta práctica se aprendió el funcionamiento del ADC para la detección de voltajes mediante una matriz cuadrada hicimos divisores de voltaje y dependiendo la tecla que oprime el usuario, este devuelve un voltaje distinto a los demás. El campo de ADC consiste de dos registros principales para poder saber el valor que tiene la entrada de analógica de forma binaria. Al principio no se sabía como poner los símbolos en el display, pero nos dimos cuenta de que el registro ADCL tiene un valor que casi no cambia como es el ADCH, por lo que nos guiamos con ese para hacer un estilo de switch como lo conocemos en otros lenguajes de programación como C o Java; para luego poner como salida el símbolo correspondiente a la tecla oprimida.

Salgado Gallegos Jesús:
Esta práctica se caracteriza por conocer el cómo detectar voltajes en el microprocesador como entrada, y generar los divisores para usarlos dependiendo de su valor. En la disposición de poner el símbolo en el display se tuvo que realizar varias comparaciones a modo de seleccionar el valor de voltaje que se ingresó, dar con el símbolo que se presiono y seleccionar el valor hexadecimal para proyectar la salida.

Sanchez Pizano Irving Daniel:
En esta práctica lo que aprendí fue el uso del ADC el cual nos sirve para detectar voltajes y así poder realizar divisores de voltaje, para eso utilizamos una matriz la cuál al presionar una tecla se iba a mostrar el resultado en un display.
También lo que se siguió utilizando fueron las interrupciones y eso nos ayudó para seguir practicando su funcionamiento.