Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Práctica 10

"Teclado Matricial"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 23/12/2022

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de realizar un teclado matricial.

Introducción Teórica

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

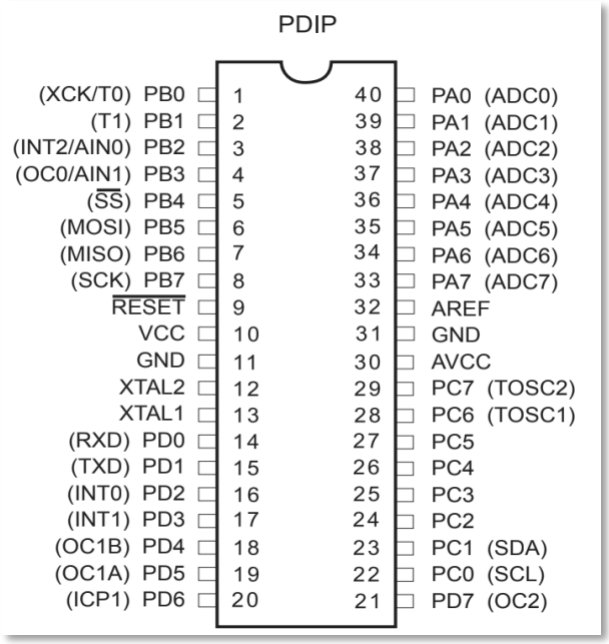

Figura 1. Configuración de pines ATmega8535.

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Teclado Matricial

Los teclados matriciales nos permiten pulsar una gran cantidad de combinaciones y emplearlos en proyectos que requieran códigos numéricos. Es un conjunto de botones agrupados, al igual que las matrices de LED, en arreglos de filas y columnas. Pese a que además de números puede contener otros caracteres como letras y otros símbolos, al fin y al cabo, lo único que generan son señales eléctricas al presionar cualquier botón.

Desde que se prende el sistema se alimenta a la matriz y al oprimir algún botón se cierra un circuito correspondiente a una fila y a una columna –como si fuera una coordenada- Por otra parte, dependiendo de las conexiones y del tipo de resistencia a emplear –ya sea pull-up o pull-down– pueden tenerse estados lógicos BAJOS al usar el pull-up o estados lógicos ALTOS.

Sin embargo, esto no sirve para identificar cuál botón se ha presionado ya que al presionarse se sabe que algún botón de cierta columna (o fila) ha cerrado el circuito, pero no se sabe a cuál fila (o columna) pertenece. Lo que debe hacerse es un barrido de filas (o columnas), esto es, que una fila quede activada mientras las otras se mantienen desactivadas por cierto lapso y después la siguiente fila queda activada y el resto se desactivan; este proceso se hace consecutivamente hasta la última fila para iniciar nuevamente con la primera fila.

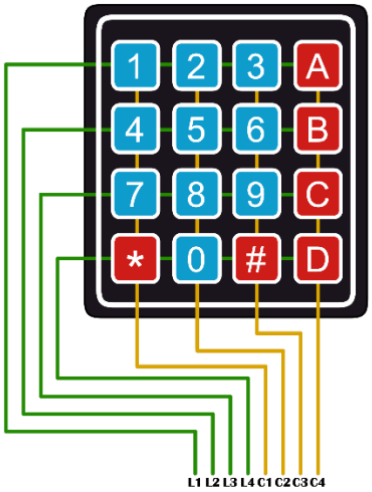

Figura 2. Imagen ilustrativa de un teclado matricial 4x4 tipo membrana.

Un teclado matricial está formado por una matriz de pulsadores dispuestos en filas (L1, L2, L3, …, LN) y columnas (C1, C2, C3, …, CN), con la intención de reducir el número de pines necesarios para su conexión. Por ejemplo, en el teclado de la figura 2 las 16 teclas necesitan sólo 8 pines del microcontrolador en lugar de los 16 pines que se requerirían para la conexión de 16 teclas independientes. Para poder leer que tecla ha sido pulsada se debe de utilizar una técnica de barrido y no solo leer un pin de microcontrolador.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 Display cátodo común
- 7 Resistores de 330 Ω a ¼ W
- 3 Resistor de 100 Ω a ¼ W
- 9 Push Button

Desarrollo Experimental

1.- Diseño de un teclado matricial de 3*3, con despliegue a display a 7 segmentos. Los pines C0, C1 y C2 serán los pines de salida por donde se enviarán los códigos de "scaneo", los pines C3, C4 y C5 serán los pines de entrada donde se leerán los botones presionados.

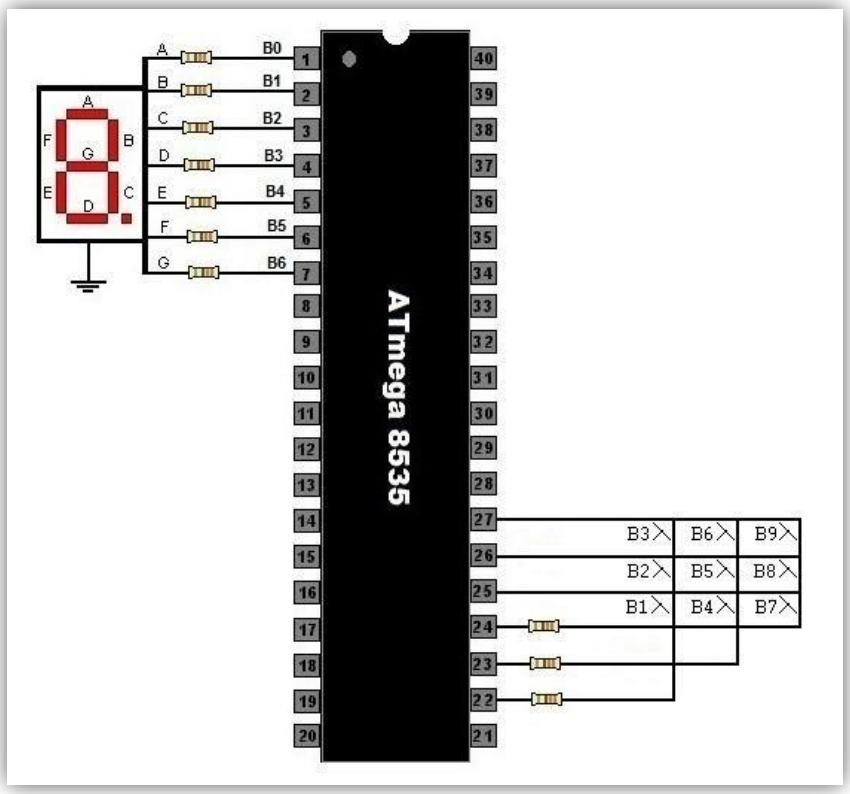

Figura 3. Circuito para el contador teclado matricial.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision.

```
#include <mega8535.h>

unsigned char tecla,lectura;

const char tabla7segmentos [10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};

void main(void)

{

// Port A initialization

DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) |
(0<<DDA1) | (0<<DDA0);
```

```
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization

DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (1<<DDB0);

// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (1<<DDC2) |
(1<<DDC1) | (1<<DDC0);

PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);

TCNT0=0x00;

OCR0=0x00;

TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;
```

```
OCR1BH=0x00;

OCR1BL=0x00;

ASSR=0<<AS2;

TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);

TCNT2=0x00;

OCR2=0x00;

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);

SFIOR=(0<<ACME);

ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);

SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1)
| (0<<SPR0);

TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1) {

//C0 A C2 son de salida y son para

//Probar las 3 columnas

//Se prueba la primera columna se envía 110

PORTC=0b00111110;

//C3, C4 y C5 son las entradas del teclado

//Por eso se enmascaran con 00111000

lectura=PINC&0b00111000;

if (lectura==0b00110000)

   tecla=7;

if (lectura==0b00101000)

   tecla=8;

if (lectura==0b00011000)
```

```
    tecla=9;
// Se prueba segunda columna se envía 101
PORTC=0b00111101;
//C3, C4 y C5 son las entradas del teclado
//Por eso se enmascaran con 00111000
lectura=PINC&0b00111000;
if (lectura==0b00110000)
    tecla=4;
if (lectura==0b00101000)
    tecla=5;
if (lectura==0b00011000)
    tecla=6;
// Se prueba tercera columna se envía 011
PORTC=0b00111011;
//C3, C4 y C5 son las entradas del teclado
//Por eso se enmascaran con 00111000
lectura=PINC&0b00111000;
if (lectura==0b00110000)
    tecla=1;
if (lectura==0b00101000)
    tecla=2;
if (lectura==0b00011000)
    tecla=3;
PORTB=tabla7segmentos [tecla];
}
}
```

Simulaciones

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

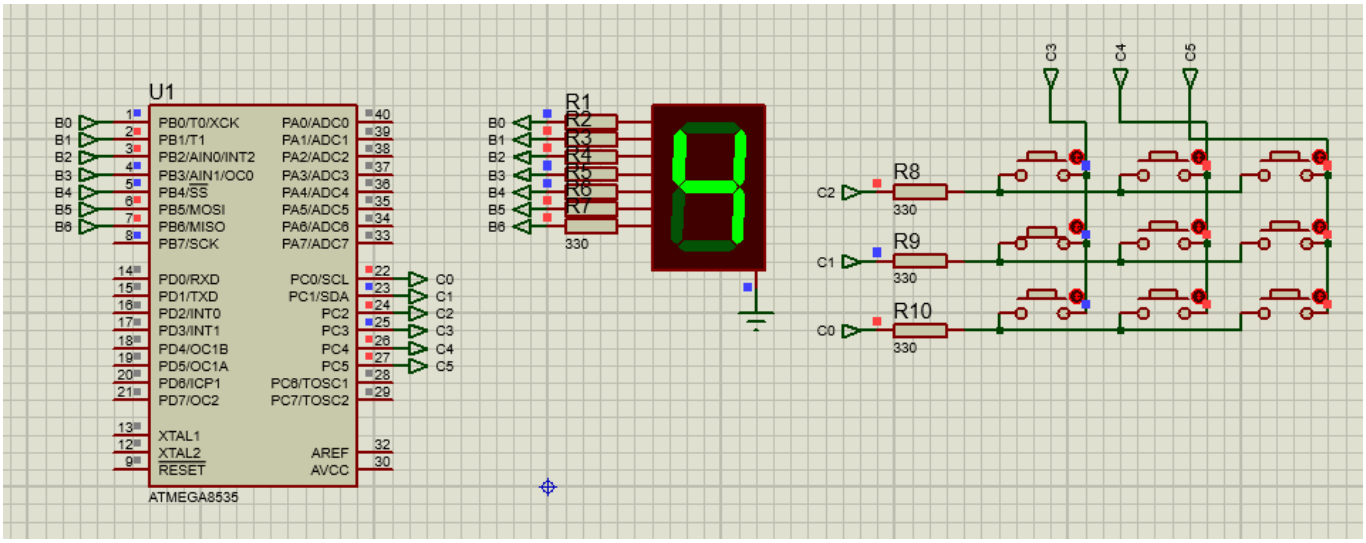

Figura 4. Simulación de la práctica 10.

Fotografía del circuito armado

A continuación, se muestra la figura 5 la evidencia sobre la realización y prueba de la practica 10.

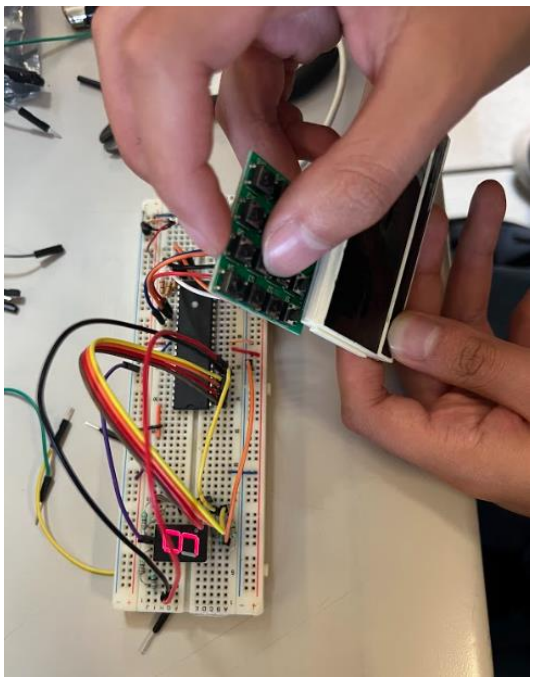

Figura 5. Circuito armado y funcionamiento de la práctica 10.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

En esta práctica a diferencia las previas, mediante el teclado matricial se obtuvo la ventaja de usar varios botones con pocas conexiones a terminales, en este caso se usaron las C y aun así quedaron algunas libres. La implementación de la práctica fue sencilla ya que el código fue proporcionado por el profesor, y el objetivo fue específicamente el de entender cómo trabaja un teclado matricial, que puede ser útil en futuras aplicaciones, del mismo modo que el código proporcionado es reusable y escalable a un teclado de mayor cantidad de filas y columnas.

Patlani Mauricio Adriana

Esta práctica no tuvo mayor complicación, solo se armó el circuito como se indicó en el diagrama y se programó con el código provisto por el profesor. En nuestro caso particular se hizo con un teclado matricial de 4x4 y se visualizó en display números. Esta práctica puede resultar útil para otras aplicaciones

Ruvalcaba Flores Martha Catalina

La practica 10 tuvo como objetivo la realización de un teclado matricial, este se realizó con uno de 4x4, y fue sencilla su implementación, puesto que el código ya estaba propuesto por el profesor y el equipo solo debió armar el circuito y probarlo. Al observar como funcionaba el teclado matricial y el resultado en el display, se entendió como funciona y cómo podemos aplicarlo en un futuro con otras prácticas.

Sandoval Hernández Eduardo

Gracias a que el código fue provisto por el profesor, esta práctica resultó sencilla de implementar, basto con armar el circuito como lo indicaba la práctica para probar su funcionamiento, esto puede resultar de gran ayuda para prácticas futuras donde se requiera utilizar una gran cantidad de botones y se disponga de pocas entradas del microcontrolador.

Bibliografía

- F. Aguilar. (2021, Mayo). 10 Teclado Matricial. Introducción a los Microcontroladores. [Online]. Disponible en: https://youtu.be/vhvTkU7--uw

- Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

- A. Camarillo. (2017). ¿Cómo funcionan los teclados matriciales y matrices de LEDs?. [Online]. Disponible en: https://blog.330ohms.com/2017/09/27/como-funcionan-los-teclados-matriciales-y-matrices-de-leds/