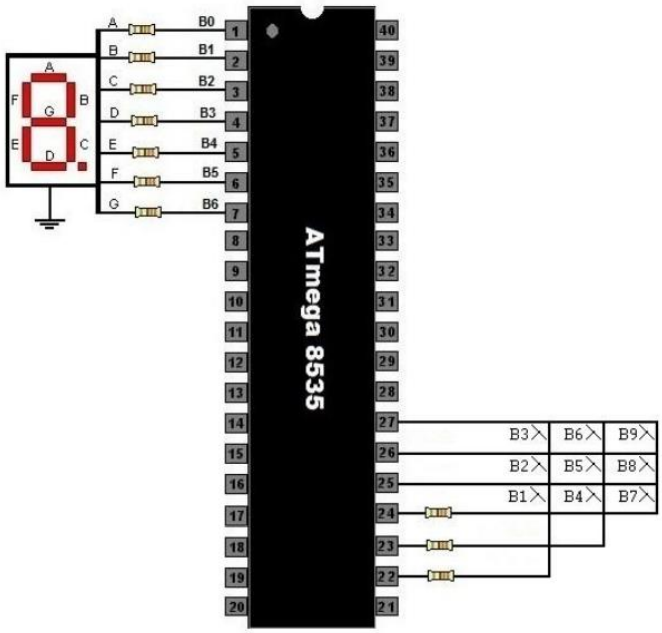
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 10

"Teclado Matricial"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

05/NOVIEMBRE/2022

I. OBJETIVO GENERAL:

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

II. MATERIAL 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.

III. INTRODUCCIÓN TEÓRICA

En esta práctica podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de integrar un teclado matricial para que se pueda observar en un Display de 7 Segmentos.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto C como entrada de nuestros botones o nuestra matriz de botones, cabe destacar en esta práctica se ocupará una Key Pad de números que es lo mismo que un arreglo de botones y el puerto B como la salida que se conectará al Display de 7 segmentos (Cátodo Común). La práctica nos mostrara el numero seleccionado de nuestros botones en el Display de 7 segmentos. En este caso de la práctica se ocuparán algunos conceptos obtenidos anteriormente en el curso como lo es la implementación de código en C para poder programar nuestro microcontrolador, algunos conceptos de Fundamentos de Diseño Digital (Display de 7 Segmentos) y de Electrónica Analógica para poder realizar correctamente el armado de nuestro circuito y que este funcione correctamente.

Ocuparemos algunos elementos como lo dice el formato de la practica ya que ocuparemos el simulador Code Vision para poder implementar el código para después programar nuestro ATMega8535, en mi caso se ocupará el programa Khazama AVR con su programador para poder programar nuestro ATMega con el archivo .hex que se generó anteriormente. Para la parte del armado del circuito será algo sencillo ya que solo se implementarán tres Display para los puertos de salida y una matriz de Botones para el puerto de entrada de nuestro circuito.

Algunos conceptos que se necesitan para poder entender algunos conceptos de la practica son los siguientes:

El KeyPad es un teclado táctil e inalámbrico que gestiona el sistema de seguridad Ajax. Se utiliza en interiores. Puede armar y desarmar estancias, informar del estado del sistema, está protegido contra adivinación de código e incluye una alarma silenciosa si coaccionan a un usuario a introducir el código.

KeyPad solo funciona con el sistema de seguridad Ajax (no se puede usar en sistemas de seguridad de otros fabricantes), al conectarse al hub mediante el protocolo de seguridad Jeweller. Rango de comunicación: hasta 1.700 metros sin obstáculos.

Esta práctica nos ayudara a entender cómo se pueden implementar los circuitos con un microcontrolador y todo el proceso que conlleva realizar la programación del microcontrolador.

Con esta introducción se podrá comprender y realizar el desarrollo de la práctica que se desarrolla más adelante.

IV. 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 "saceneo", los pines C3, C4 y C5 serán los pines de entrada donde se leerán los botones presionados.

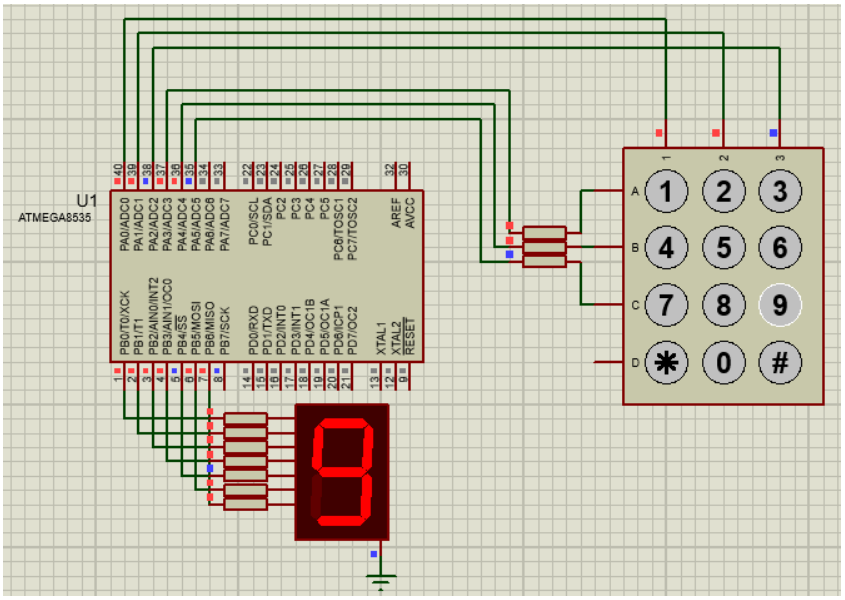


Fig.1_*Circuito Armado y Simulado en Proteus.*

CIRCUITO EN FISICO ARMADO.

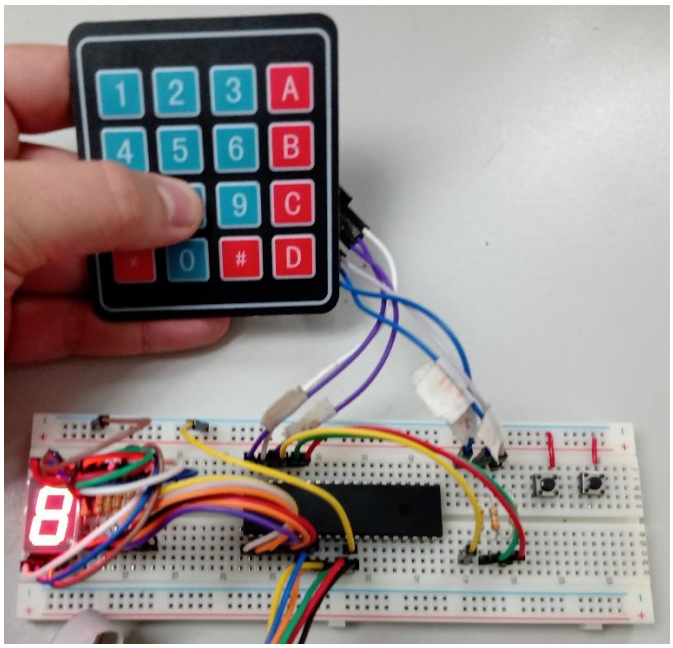


Fig.2_*Circuito Armado en Físico (Teclado Matricial).*

V. CÓDIGO

```
#include <mega8535.h>
#include <delay.h>

// Declare your global variables here
unsigned char tecla, lectura;
const char tabla7segmentos [10] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=Out Bit1=Out Bit0=Out
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (1<<DDA2) | (1<<DDA1) |
(1<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=1 Bit1=0 Bit0=0
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (1<<PORTA2) |
(0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
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
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
(0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
(0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) |
(0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
(0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
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;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8)
| (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
```

```
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);

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

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

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
      {
        tecla = 0;
        PORTA=0b00111110;
        lectura=PINA&0b00111000;
        if (lectura==0b00110000) tecla=1;
        if (lectura==0b00101000) tecla=4;
        if (lectura==0b00011000) tecla=7;

        PORTA=0b00111101;
        lectura=PINA&0b00111000;
        if (lectura==0b00110000) tecla=2;
        if (lectura==0b00101000) tecla=5;
        if (lectura==0b00011000) tecla=8;

        PORTA=0b00111011;
        lectura=PINA&0b00111000;
        if (lectura==0b00110000) tecla=3;
        if (lectura==0b00101000) tecla=6;
        if (lectura==0b00011000) tecla=9;

        PORTB=tabla7segmentos [tecla];
        delay_ms(50);
      }
```

VI. OBSERVACIONES Y CONCLUSIONES

Gracias a la realización de la práctica podemos observar cómo se utiliza e implementa un circuito con un microcontrolador (ATMega8535) y como gracias a su programador pudimos logra programar el integrado con nuestro código en lenguaje C. En esta práctica al momento de armar el circuito y simularlo pudimos observar cómo se implementaría un Teclado Matricial, que realmente no fue tan difícil.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar y programar el ATMega8535. También al observar el data set del Display se comprendido mejor como conectarlo en físico este componente.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que preferí utilizar Key Pad que hacer mi propio arreglo Matricial con Botones, que será lo mismo. Gracias a la anterior practica ya fue más fácil de hacerlo, ya que es casi lo mismo que esta.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2022, de CodeVisionAVR Sitio web: http://www.hpinfotech.ro/cvavr-download.html

- ✓ Ajax Systems Manufacturing. (2022). Manual de usuario del KeyPad. 05/Noviembre/2022, de Ajax, Sitio web: https://support.ajax.systems/es/manuals/keypad/#:~:text=KeyPad%20es%20un%20teclado%20t%C3%A1ctil%20que%20controla%20el,fijo%20que%20se%20ubica%20dentro%20de%20una%20estancia.