INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA 5 y 6

"CONTADOR ACTIVADO POR FLANCOS"
“CONTADOR SON REBOTES"

SISTEMAS EN CHIP

6CM3

PROFESOR:

AGUILAR SÁNCHEZ FERNANDO

INTEGRANTES:

MIGUEL VÁSQUEZ MARIANO JOSUÉ

FRANCO OLVERA ODER

GONZALEZ RAMOS ANGEL DAVID

Fecha de Elaboración

13/03/2024

INTRODUCCIÓN

En el ámbito de los sistemas digitales, la implementación de contadores activados por flancos y contadores con eliminación de rebotes es esencial para numerosas aplicaciones donde se requiere un conteo preciso y confiable de eventos. Estos tipos de contadores son fundamentales en sistemas de control, monitoreo, temporización y muchas otras áreas donde la precisión y la estabilidad son críticas.

Un contador activado por flancos es aquel que incrementa o decrementa su conteo cuando detecta un cambio en el estado de una señal de entrada, ya sea un flanco de subida o un flanco de bajada. Por otro lado, un contador con eliminación de rebotes es aquel que incorpora mecanismos para filtrar y eliminar las fluctuaciones transitorias en la señal de entrada, garantizando así un conteo preciso y sin errores causados por estos rebotes.

En esta práctica, exploraremos los principios teóricos y prácticos detrás de los contadores activados por flancos y los contadores con eliminación de rebotes. Analizaremos cómo se aplican estos conceptos en el diseño de sistemas digitales y cómo se implementan en la práctica utilizando diversas técnicas y tecnologías.

Al comprender a fondo estos aspectos, los ingenieros y diseñadores de sistemas digitales podrán desarrollar soluciones más robustas y confiables que requieran funcionalidades de conteo en sus aplicaciones, garantizando así un rendimiento óptimo y una precisión adecuada en la medición y control de eventos.

CÓDIGO

Contador flancos

```
/*****************************************************
This program was created by the CodeWizardAVR V4.01
Automatic Program Generator
© Copyright 1998-2024 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 19/02/2024
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>

#define boton PIND.0

bit botonp;

bit botona;

unsigned char var;

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


// Declare your global variables here


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=In Bit1=In Bit0=In

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

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

```
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<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=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P

PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<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)

      {

          if (boton==0)

            botona=0;

          else

            botona=1;


          if ((botona==0) && (botonp==1))

            var++; //Se incrementa la variable


          if (var==10)

            var=0;


          PORTB=tabla7segmentos [var];

          botonp=botona;

      }

}
```

Contador sin rebote

```
#include <mega8535.h>

#include <delay.h>

#define boton PIND.0
```

```
bit botonp;

bit botona;

unsigned char var;

const char tabla7segmentos
[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,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=In Bit1=In Bit0=In

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

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

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

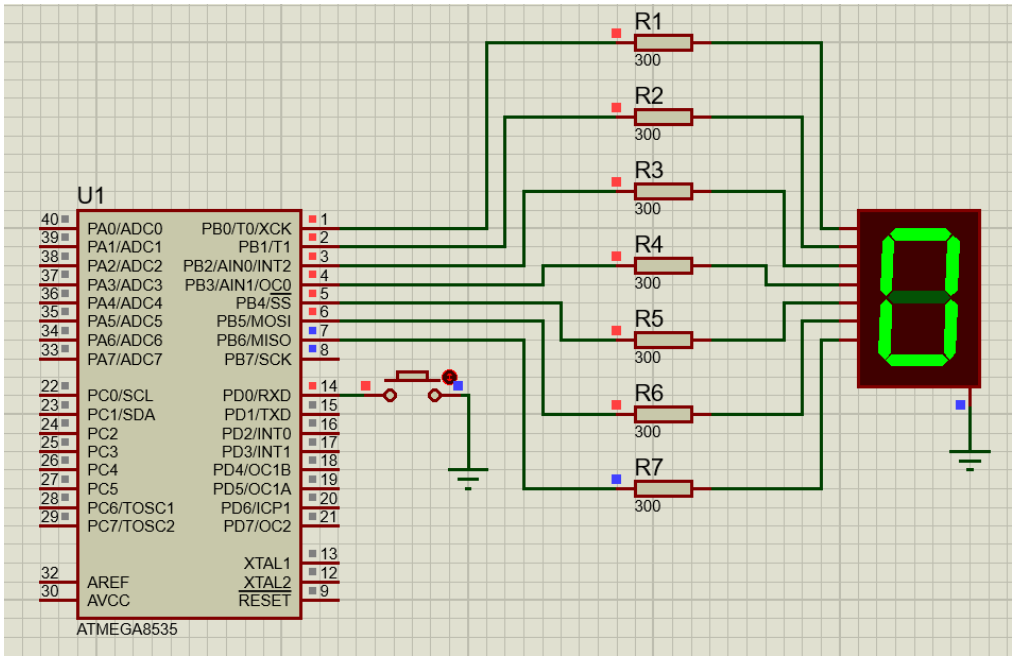
```
// Port D initialization

// 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=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P

PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<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;
```

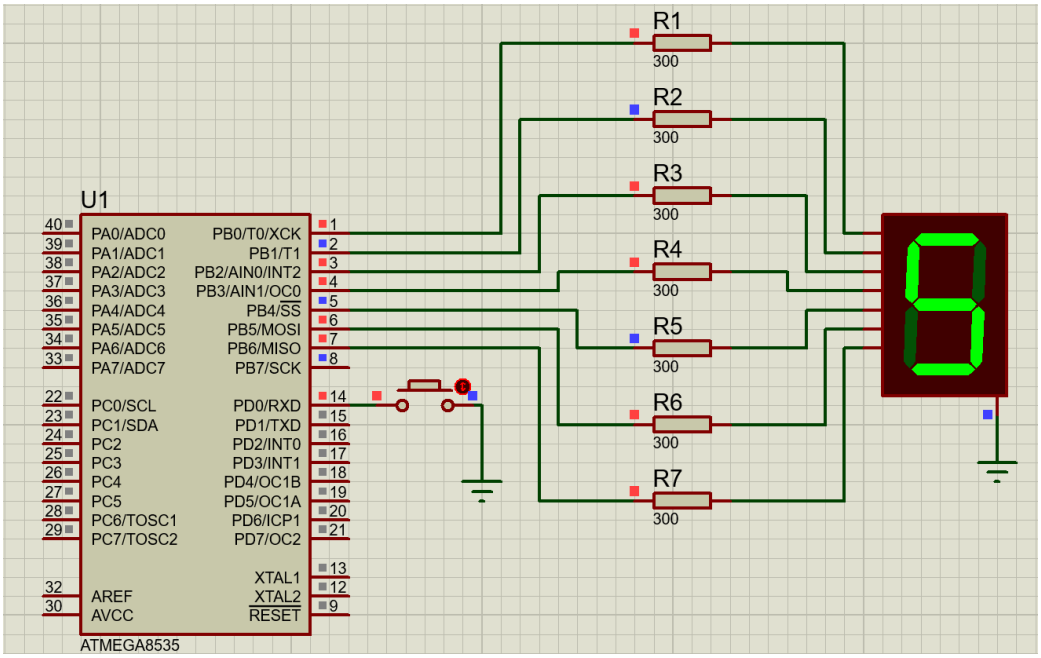
```
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// 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)
      {
        if (boton == 0)
            botona = 0;
        else
            botona = 1;


        if ((botona == 0) && (botonp == 1)) {
            var++;
```

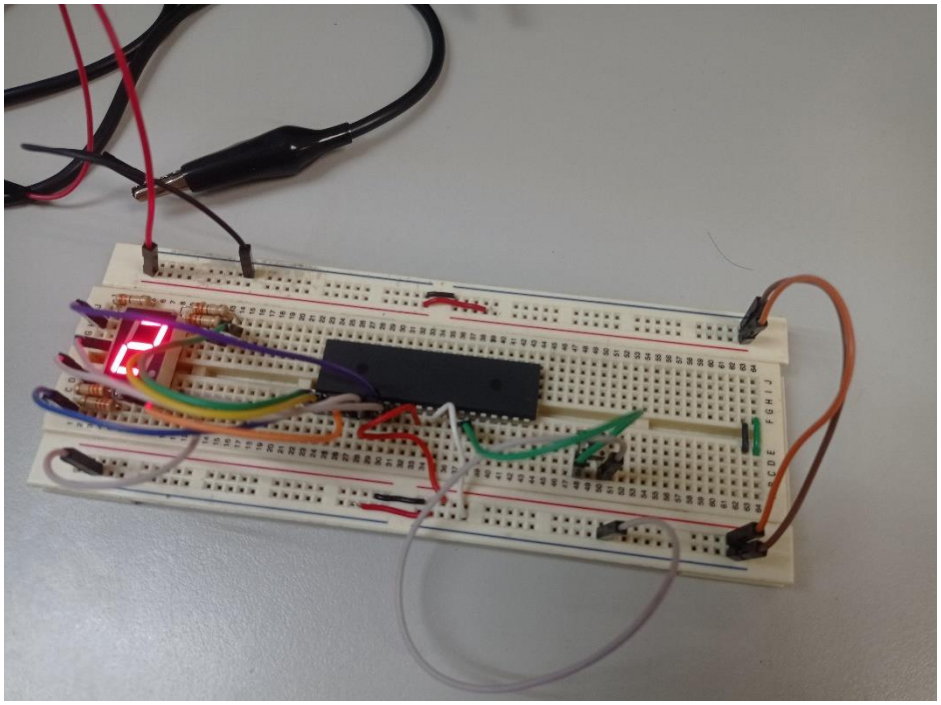
```
            if (var == 10)
                var = 0;

            delay_ms(40);
          }

          if ((botonp == 0) && (botona == 1))
            delay_ms(40);

          PORTB = tabla7segmentos[var];
          botonp = botona;
        }
}
```

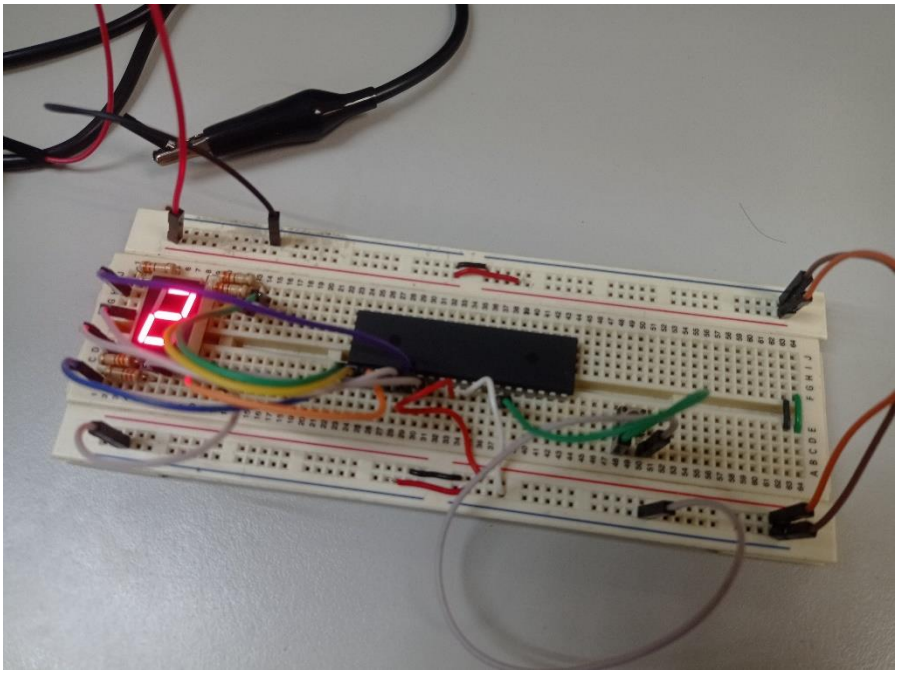
SIMULACIÓN EN PROTEUS




R1
300
R2
300
R3
300
R4
300
R5
300
R6
300
R7
300
U1
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
ATMEGA8535

FOTOS





CONCLUSIONES INDIVIDUALES

Franco Olvera Demian Oder

Hay situaciones cuando no se presta atención a la salida y el medio por el que esta se proporciona en un sistema; la práctica previa es un ejemplo de este caso. En este caso, se aplicaron dos soluciones para intentar mitigar el problema ocurrido en la práctica 4: una activación por flancos y otra sin rebotes. La activación por flancos parece resolver el problema en un todo, pero no completamente; existe un barrido ocasional que desplazaba el número del Display, haciendo que, en algunos casos, éste avanzara más de una vez. Por otro lado, la activación sin rebotes es la solución correcta en este caso ya que permitía que el Display se moviese adecuadamente. De esta manera, queda demostrado que el problema no es la arquitectura o el circuito, sino que la programación juega un papel igual de importante.

Gonzalez Ramos Angel David

La práctica de implementar un contador activado por flancos y un contador con rebotes en un microcontrolador como el ATmega8535 ofrece una comprensión profunda de la detección y gestión de eventos de entrada. Un contador activado por flancos registra cambios específicos en el estado de una señal de entrada, como flancos de subida o bajada, para contar eventos precisos. Por otro lado, un contador con rebotes aborda los problemas causados por fluctuaciones transitorias en la señal

de entrada, asegurando que solo se cuenten cambios válidos después de que la señal se estabilice. Estas prácticas son fundamentales para el diseño robusto de sistemas que requieren precisión en la detección de eventos y resistencia a interferencias eléctricas.

Miguel Vásquez Mariano Josué

En la practica 5, pudimos resolver el problema que presentaba el contador de la practica 4 cuando presionábamos el botón. Esto se logró mediante el uso variables auxiliares que se activaban y desactivaban cuando se presionaba y se dejaba de presionar el botón. Sin embargo, aun persistía un pequeño problema causado por la manera en cómo funciona el push button, ya que estos a la hora de soltarse presentan un pequeño rebote que ocasionaban que realizaran un conteo de mas a la hora de soltar el botón. En la practica 6 se pudo lograr resolver este problema simplemente agregando delays en las comprobaciones de los estados auxiliares, con el objetivo de que detener al programa por un pequeño lapso hasta que el rebote ocurriera.

REFERENCIAS

- Arpita, M. D. (2012, 14 enero). CONTADOR DIGITAL DE 0 a 9 CON 7490. Mikitronic Tutoriales Electrónica. Recuperado 28 de marzo de 2024, de https://mikitronic.blogspot.com/2012/01/contador-digital-de-0-9.html
- Vega, E. (2017, 7 enero). Pulsadores y antirrebote con Arduino. Arduino para todos. Recuperado 28 de marzo de 2024, de https://arduparatodos.blogspot.com/2017/01/pulsadores-y-antirrebote-con-arduino.html