EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MORELIA
"José María Morelos y Pavón"

TECNOLÓGICO NACIONAL DE MÉXICO CAMPUS MORELIA

Reporte

MOTOR

PRESENTA:

Cristian David Nuñez Cambron

Santiago Gonzalez Lara

PROFESOR:

Ing .Jorge Luis Diaz Huerta

MORELIA, MICHOACÁN

INTRODUCCION

Este reporte describe la implementación de un sistema de control para LEDs utilizando un microcontrolador AVR (específicamente el ATmega328P), donde se manejan entradas digitales para incrementar, decrementar y alternar estados de salida. El programa demostrado es un ejemplo práctico de cómo interactuar con periféricos de hardware a bajo nivel, manejando tanto entradas como salidas digitales.

El sistema implementado tiene las siguientes características principales:

1. Entradas Digitales:

- Utiliza los pines del Puerto B (PB0-PB2) para detectar pulsaciones que incrementan un contador y alternan salidas
- Emplea el pin PD6 como entrada para decrementar el contador

2. Salidas Digitales:

- Controla LEDs conectados a los pines PD2-PD5 (4 bits) que muestran el valor actual de un contador de 4 bits (0-15)
- Maneja dos LEDs adicionales en los pines PC2 y PC3 que se alternan con una pulsación específica

3. Características Implementadas:

- Lógica antirrebote (debounce) para todas las entradas
- Contador con límite superior (15) e inferior (0)
- Visualización directa del valor del contador en 4 LEDs
- Mecanismo de alternancia entre dos LEDs con una entrada específica

Relevancia del Tema

Este tipo de implementación es fundamental en sistemas embebidos y constituye la base para:
- Sistemas de control industrial
- Interfaces hombre-máquina (HMI) simples
- Mecanismos de visualización de estados
- Prototipado de sistemas interactivos

El código analizado muestra técnicas esenciales como:
- Configuración de puertos de entrada/salida
- Manejo de retardos para antirrebote
- Manipulación de bits individuales
- Implementación de máquinas de estados simples
- Técnicas de polling para lectura de entradas

En el siguiente reporte se analizará en detalle cada componente del sistema, su implementación y los principios teóricos que lo sustentan.

CIRCUITO

En este aspecto se tomo en cuenta con el codigo el circuito debe hacer lo siguiente

1. Se debe presionar el primer boton para que se enciendan los leds y empiece a contar hasta 15 que debe ser el maximo y debera no pasar de conteo
2. El sensor a la hora de pasar un objeto para que lo detecte debe decrementar el numero que este en los leds y si es 0 debe bloquearse y no reiniciar el contador
3. El tercer boton lo que hace es encender los leds de la otra parte de la protoboard indicando una direccion si lo presionas una vez enciende refiriendose que va a la derecha y si lo vuelves a presionar se apaga el otro y enciende el otro led marcando que va a la izquierda

El siguiente video es su funcionamiento

Video de WhatsApp 2025-06-03 a las 20.09.50_952bc102.mp4

CODIGO

```
Config:

    LDI R16, 0b00000000
    OUT DDRB, R16
    LDI R16, 0b00000111
    OUT PORTB, R16


    LDI R16, 0b00001100
    OUT DDRC, R16
    LDI R16, 0b00000000
    OUT PORTC, R16


    LDI R16, 0b00000000
    OUT DDRD, R16
    LDI R16, 0b01000000
    OUT PORTD, R16


    LDI R16, 0b00111100
    OUT DDRD, R16

    LDI R18, 0x00
    LDI R19, 0b00000111
    LDI R25, 0b00000001
    LDI R26, 0b00000100
```

Se configura los valores que tedran los Registros en los puertos b y d, haciendo tambien PULLUPS en el R16

Registros Clave

- R18: Valor del contador (0–15)
- R19: Estado anterior de botones PB0–PB2
- R25: Estado anterior del botón en PD6

```
MainLoop:

    IN R21, PINB
    ANDI R21, 0b00000111


    CP R21, R19
    BREQ CheckPD6


    RCALL DebounceDelay


    IN R22, PINB
    ANDI R22, 0b00000111


    CP R22, R21
    BRNE CheckPD6


    MOV R19, R22
```

El bloque principal del programa es un bucle infinito llamado MainLoop, donde se gestionan continuamente las entradas del usuario (botones) y se actualiza el comportamiento del sistema en función de ellas. A continuación, se describen los pasos con mayor profundidad:

- Se leen los 3 bits menos significativos del puerto PINB, correspondientes a los botones conectados a PB0, PB1 y PB2.
- Se almacenan en el registro R21.
- Se descartan los demás bits con ANDI, dejando solo PB0–PB
- Si no hubo cambio (R21 = R19), se omite el procesamiento de botones y se salta a la verificación de PD6.
- Si hubo un cambio, se llama a una rutina de antirrebote (RCALL DebounceDelay), y se vuelve a leer el estado de los botones para confirmar el cambio real (no por ruido eléctrico).
- Después del segundo chequeo, si el estado sigue siendo distinto, se actualiza R19 con el nuevo estado (MOV R19, R22), y se procede a interpretar los botones activos.

```
MOV R20, R22
ANDI R20, 0b00000001
CPI R20, 0b00000000
BRNE CheckPB2
RCALL Incrementar
RCALL MostrarEnLEDs
```

- Se verifica si PB0 (bit 0) está en nivel bajo (0), lo cual indica que el botón está presionado (por el pull-up).

- Si es así, se llama a la subrutina Incrementar que aumenta el valor del contador (R18) en una unidad (máximo 15).

- Luego se llama a MostrarEnLEDs, que actualiza los LEDs en PD2–PD5 para reflejar el nuevo valor.

```
CheckPB2:

    MOV R20, R22
    ANDI R20, 0b00000100
    CPI R20, 0b00000000
    BRNE CheckPD6
    RCALL AlternarPC2y3
```

- Verifica si PB2 (bit 2) está presionado.

- Si lo está, se llama a la subrutina AlternarPC2y3, la cual alterna el encendido entre PC2 y PC3 (posiblemente indicando diferentes estados o modos del sistema).

```
CheckPD6:
    ; --- Procesar PD6 para decrementos ---
    IN R23, PIND
    ANDI R23, 0b01000000


    LDI R20, 0b01000000
    AND R23, R20
    LDI R20, 0b00000000
    CPI R23, 0b01000000
    BREQ PD6High
    LDI R20, 0b00000001
PD6High:
    MOV R23, R20

    ; Verificar si hubo cambio
    CP R23, R25
    BREQ MainLoop

    ; Esperar debounce
    RCALL DebounceDelay

    ; Leer nuevamente
    IN R24, PIND
    ANDI R24, 0b01000000

    ; Mismo proceso de conversión
    LDI R20, 0b01000000
    AND R24, R20
    LDI R20, 0b00000000
    CPI R24, 0b01000000
    BREQ PD6High2
    LDI R20, 0b00000001
```

```
PD6High2:
    MOV R24, R20


    CP R24, R23
    BRNE MainLoop


    MOV R25, R24

    CPI R24, 0b00000001
    BREQ MainLoop

    RCALL Decrementar
    RCALL MostrarEnLEDs

    RJMP MainLoop
```

- Se lee el bit 6 del puerto D y se compara con el valor anterior (R25).
- Si no hubo cambio, se regresa al inicio del bucle.

Debounce y segunda lectura

- Si se detecta un cambio:
- Se espera con DebounceDelay.
- Se vuelve a leer el estado de PD6 (R24).
- Se compara con la primera lectura para confirmar la pulsación.

```
Incrementar:
    CPI R18, 15
    BRGE NoIncrementar
    INC R18
    ANDI R18, 0x0F     ;
```

- Si el botón ya fue liberado (transición alta a baja confirmada), se ejecuta Decrementar, que reduce el valor de R18 en uno (mínimo 0).
- Luego, MostrarEnLEDs actualiza los LEDs para mostrar el nuevo valor.

Esta subrutina disminuye el valor del contador R18, asegurándose de que nunca baje de 0 y se mantenga dentro del rango de 4 bits (0 a 15):

```
Decrementar:
    CPI R18, 0
    BREQ NoDecrementar
    DEC R18
    ANDI R18, 0x0F
```

- CPI y BREQ aseguran que no se decrementa si ya se llegó a 0.

- ANDI R18, 0x0F limpia los bits altos en caso de que accidentalmente hayan sido modificados por otras operaciones, garantizando un valor entre 0 y 15 (4 bits).

Subrutina que alterna el estado de los pines PC2 y PC3, utilizados como salidas digitales (por ejemplo, para controlar LEDs):

```
AlternarPC2y3:
    IN R16, PORTC
    LDI R20, 0b00001100
    AND R16, R20
    CPI R16, 0b00000000
    BREQ EncenderPC2
    CPI R16, 0b00000100
    BREQ EncenderPC3
    CPI R16, 0b00001000
    BREQ EncenderPC2
    RET


EncenderPC2:
    IN R16, PORTC
    ANDI R16, 0b11110011
    ORI R16, 0b00000100
    OUT PORTC, R16
    RET
```

Simula un "toggle" entre dos salidas, útil para indicadores visuales o control de modo (modo A / modo B, por ejemplo).

```
MostrarEnLEDs:

    LDI R16, 0b00000000

    MOV R20, R18
    ANDI R20, 0b00000001
    CPI R20, 0b00000001
    BRNE CheckBit1
    ORI R16, 0b00000100

CheckBit1:

    MOV R20, R18
    ANDI R20, 0b00000010
    CPI R20, 0b00000010
    BRNE CheckBit2
    ORI R16, 0b00001000

CheckBit2:

    MOV R20, R18
    ANDI R20, 0b00000100
    CPI R20, 0b00000100
    BRNE CheckBit3
    ORI R16, 0b00010000

CheckBit3:

    MOV R20, R18
    ANDI R20, 0b00001000
    CPI R20, 0b00001000
    BRNE OutputLEDs
    ORI R16, 0b00100000

OutputLEDs:
    OUT PORTD, R16
    RET
```

- Si R18 = 5 (0b0101), se encenderán los LEDs en PD2 y PD4 (bit 0 y bit 2 en 1).
- Esto proporciona una representación visual en binario del contador.

```
DebounceDelay:
    LDI R23, 160
DebounceLoop1:
    LDI R24, 200
DebounceLoop2:
    DEC R24
    BRNE DebounceLoop2
    DEC R23
    BRNE DebounceLoop1
    RET
```

Una rutina de espera controlada por software, usada para evitar errores por rebotes eléctricos al presionar botones físicos (rebotes mecánicos o eléctricos):

- Se ejecuta un doble bucle anidado.
- El tiempo total de espera depende de los valores cargados en R23 y R24. En este caso, es aproximadamente 160 × 200 ciclos de bucle, una espera corta suficiente para eliminar rebotes.

CONCLUSION

Este programa en lenguaje ensamblador para el microcontrolador ATmega328PB demuestra el uso efectivo de registros, máscaras de bits y control de puertos para implementar una interfaz básica de entrada/salida. A través de botones conectados al puerto B y al pin PD6, se permite al usuario **incrementar o decrementar un contador** de 4 bits almacenado en el registro R18. Este valor se **visualiza dinámicamente en un conjunto de LEDs conectados al puerto D (PD2–PD5)**, proporcionando una representación binaria clara e intuitiva.

Además, se incorpora una subrutina de **alternancia de pines PC2 y PC3**, lo cual permite activar indicadores secundarios o controlar modos de operación. También se incluye una **rutina de antirrebote (debounce)** para garantizar lecturas confiables de los botones, aumentando la robustez del sistema frente a fluctuaciones eléctricas comunes en hardware físico.

En conjunto, el programa cumple con los principios fundamentales de diseño embebido: control eficiente del hardware, gestión de entradas digitales con validación, retroalimentación visual al usuario y un flujo de control robusto basado en estados. Este proyecto representa una base sólida para desarrollos más avanzados en microcontroladores, tales como menús interactivos, temporizadores, o sistemas de conteo con múltiples entradas y salidas.