

DOCUMENTACIÓN 2

Flujo Completo de Diseño, Simulación e
Implementación de Firmware para un ATmega328P

Autor: Luis David Barahona Valdivieso

Fecha: 18/04/2025

ÍNDICE

1. Planteamiento del problema e identificación de requerimientos.....	3
1.1. Problema.....	3
1.2. Requerimientos (periféricos y funciones).....	3
2. Propuesta de solución (etapa de diseño).....	3
2.1. Diagrama de bloques.....	3
2.2. Configuración de pines asumida.....	3
2.3. Diagrama de flujo.....	4
2.4. Código en ensamblador AVR.....	4
2.5. Código en C.....	6
3. Simular la propuesta de solución (etapa de simulación).....	6
3.1. Simulación en Microchip Studio.....	6
3.1.1. Crear proyecto en Microchip.....	6
3.1.2. Iniciar simulación.....	7
3.1.3. Análisis de registros.....	8
3.1.4. Análisis de puertos I/O.....	8
3.1.5. Análisis del registro de banderas o STATUS REGISTER.....	9
3.1.6. Validación de casos.....	10
3.1.6.1. Evidencia de configuración adecuada de puertos I/O.....	10
3.1.6.2. Evidencia de carga del número 1.....	11
3.1.6.3. Evidencia de carga del número 2.....	11
3.1.6.4. Evidencia de la suma del número 1 y número 2.....	12
3.1.6.5. Evidencia de la resta del número 1 número 2.....	12
3.2. Simulación en Proteus 8 Professional.....	13
4. Prototipar la propuesta de solución (etapa de prototipado).....	13
4.1. Diagrama de conexiones.....	13
4.2. Validación de casos.....	13
5. Implementar la propuesta de solución (etapa de diseño de PCB y fabricación).....	13
5.1. Diseño de componentes.....	13
5.2. Diseño de esquemático.....	13
5.3. Diseño de PCB.....	13
5.4. Generación de BOM.....	13
5.5. Generación de archivos de fabricación.....	13
6. Documentación en Github.....	13
7. Anexos.....	13
7.1. Páginas web adicionales.....	13
7.2. Palabras reservadas dentro de Ensamblador AVR (ATmega328P).....	14
7.3. Lista de instrucciones lógicas, aritméticas y de salto usadas en el ejercicio de ejemplo.....	15
8. Otros.....	15

1. Planteamiento del problema e identificación de requerimientos

1.1. Problema

Implemente un programa en lenguaje **ensamblador** que utilice los switches SW3:SW0 para ingresar 2 números de 4 bits, siendo SW3 el MSB.

Al presionar el switchr B3, el primer número se deberá guardar en el registro R19 y visualizar en el nibble menos significativo de los LEDs L7 – L0. Cuando se presione el switchr B2, se deberá guardar el segundo número en el registro R20 y mostrarlo en el nibble más significativo, al lado del primer número. Finalmente, cuando se presione B1 se debe realizar la suma de los 2 números y visualizar el resultado de la suma en los LEDs; y cuando se presione B0, se debe visualizar el resultado de la resta. Si el resultado de la resta es negativo, debe indicarse mediante el encendido del LED 9. Si el resultado de la operación es cero, debe indicarse encendiendo el LED 10. El resultado debe visualizarse sobre los LEDs, hasta que se vuelva a presionar el switchr B2 o B3 para ingresar un nuevo número, y repetir el proceso sin necesidad de reiniciar el microcontrolador.

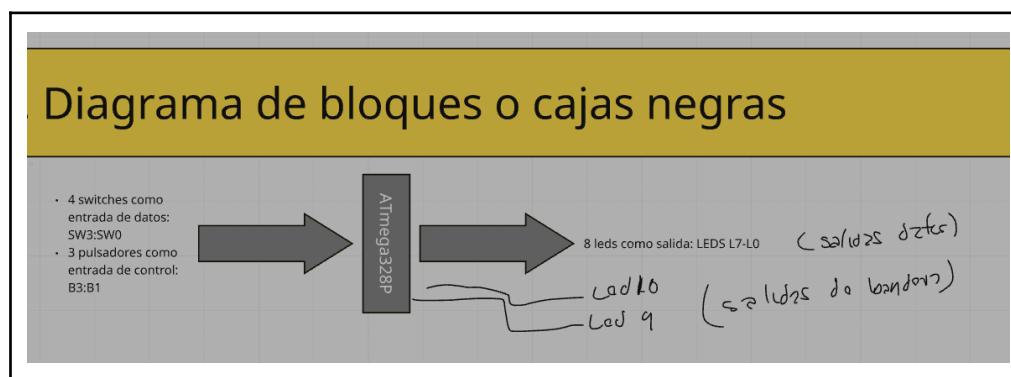
1.2. Requerimientos (periféricos y funciones)

- **SW3–SW0:** switches para ingresar un número de 4 bits (SW3 = MSB).

- **switchr B3:** guarda número en R19 y lo muestra en LEDs L3–L0 (nibble LSB).
- **switchr B2:** guarda número en R20 y lo muestra en LEDs L7–L4 (nibble MSB).
- **switchr B1:** realiza suma de R19 + R20 → muestra en L7–L0.
- **switchr B0:** realiza resta R19 – R20, muestra en L7–L0.
 - Si negativo → enciende LED9.
 - Si cero → enciende LED10.
- **Entradas configuradas como pull-down**

2. Propuesta de solución (etapa de diseño)

2.1. Diagrama de bloques



2.2. Configuración de pines asumida

PINES (en base a cualquiera de los 3 puertos del ATmega328P)

PC3-PC0 (leen el estado de los switches)	SW3-SW0 (Inputs as a pull down)
PB3-PB0 (leen el estado de los switchres)	B3-B0 (Inputs as a pull down)
PC5-PC4 (muestran/setean/escriben el estado de las banderas)	Led10-Led9 (Output as a flag)
PD7-PD0 (muestran la data)	LEDs L7-L0 (Outputs as a data)

1. PUERTO B

- Registro DDRB: Permite configurar si el puerto será de entrada(0) o salida(1).
- Registro PINB: Siempre está leyendo el estado de los 8 bits del puerto B.

- Registro PORTB: Permite setear/resetear salidas. Permite mostrar datos. Se usa, solo cuando el puerto B ha sido configurado como salida.

2. PUERTO C

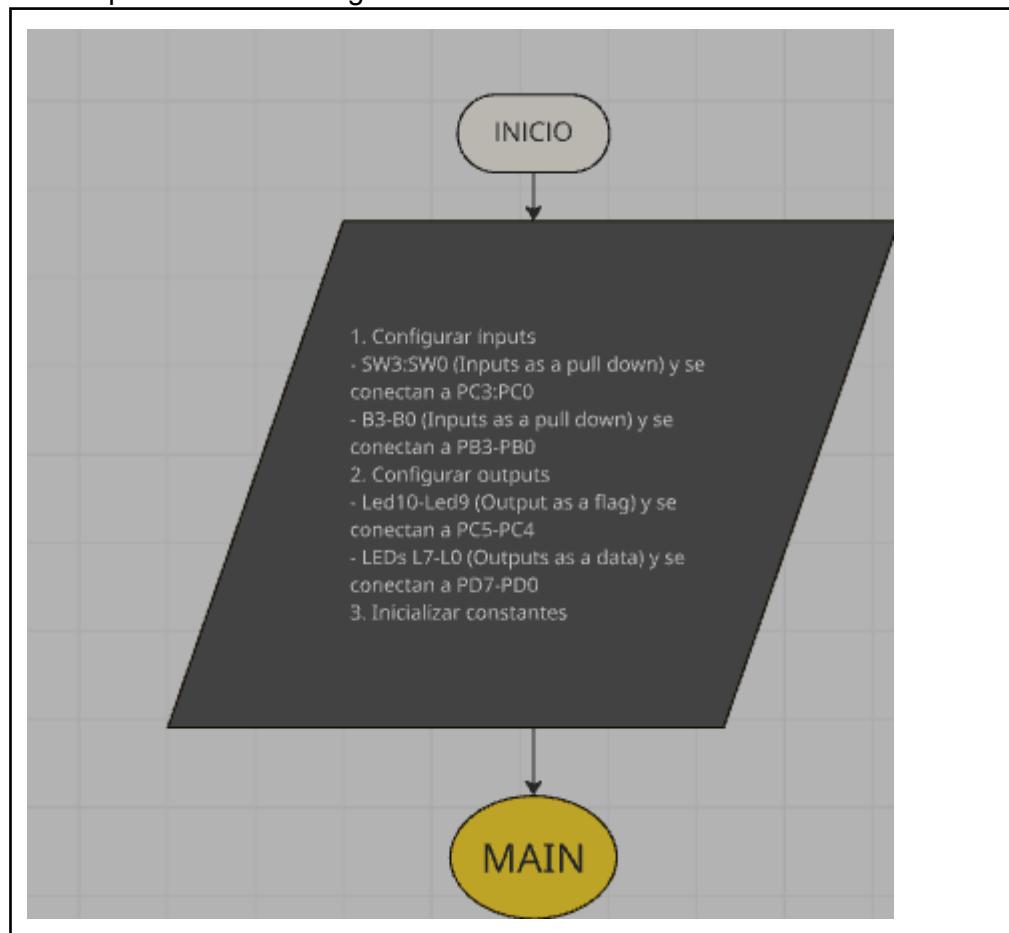
- Registro DDRC: Eset permite configurar los 8 bits del puerto C como entrada o salida.
- Registro PINC: Este siempre lee el estado de los 8 bits del puerto C.
- Registro PORTC: Si ha sido configurado como salida. Entonces, es usado para escribir.

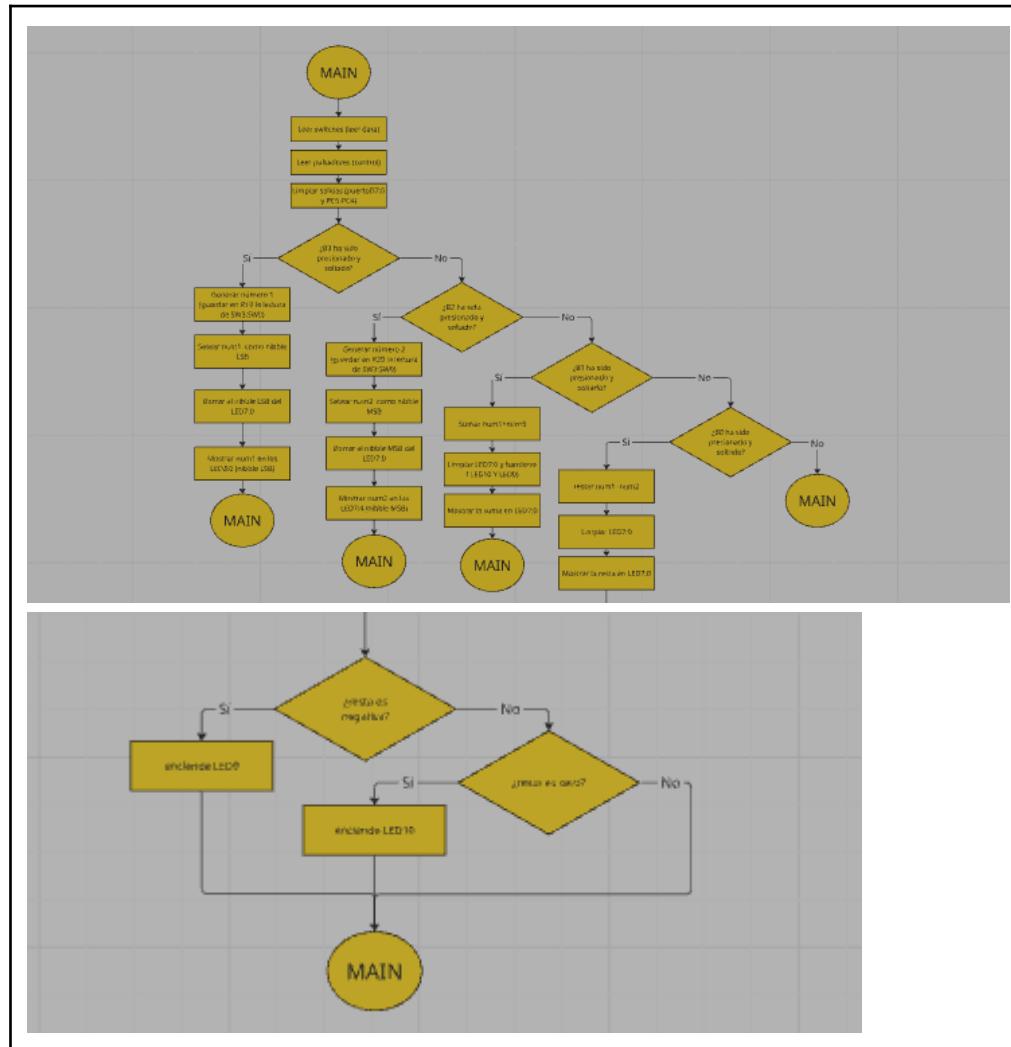
3. PUERTO D

- Registro DDRD
- Registro PIND
- Registro PORTD

2.3. Diagrama de flujo

Se usa para describir la lógica de funcionamiento del firmware.





2.4. Código en ensamblador AVR

```

; Retol.asm
;
; Created: 18/04/2025 07:28:00
; Author : luisdavidbarahona
;

; Replace with your application code
; --- Directivas iniciales ---
.include "m328Pdef.inc"

.org 0x00
rjmp RESET

; --- Variables temporales ---

```

```
.def temp = r16
.def input = r17
.def result = r18
.def num1 = r19
.def num2 = r20
.def diff = r21

; --- RESET vector ---
RESET:
    ; Configurar switches (PD3-PD0) como entrada
    cbi DDRC, 0
    cbi DDRC, 1
    cbi DDRC, 2
    cbi DDRC, 3
    ; Configurar entradas para que sean pull down
    cbi PORTC, 0 ; El PC0 =0 => esto me permite crear un
pull down fisico, fuera del atmega
    cbi PORTC, 1 ; El PC1=0 => same
    cbi PORTC, 2 ; El PC1=0 => same
    cbi PORTC, 3 ; El PC1=0 => same

    ; Configurar switchres (PB0-PB3) como entrada
    cbi DDRB, 0
    cbi DDRB, 1
    cbi DDRB, 2
    cbi DDRB, 3

    ; Configurar switchres como pull down = desactivar
pull up
    cbi PORTB, 0 ; el pull down será creado fuera del
atega usando una resistencia y un switchr
    cbi PORTB, 1 ; PORTB[1]=0
    cbi PORTB, 2
    cbi PORTB, 3

    ; No pull-up, pull-down externo (no se configura en el
AVR, se asume por hardware externo)

    ; Configurar PORTC como salida para PC5 para LED9 y
PC4 para LED10
    sbi DDRC, 5 ; DDRC[5]=1
    sbi DDRC, 4 ; DDRC[4]=1

    ; Configurar PORTD (PD7-PD0) como salida (LEDs L7-L0)
    ldi temp, 0xFF
    out DDRD, temp

    ; Limpiar LEDs (los inicializamos apagados)
    ldi temp, 0x00
    out PORTD, temp ; PORTD = 0x00 = 0b0000 00000
    cbi PORTC, 5 ; Limpia el PORT[5]=0 LIMPIA EL LED 10
```

```
        cbi PORTC, 4 ; Limpia el LED 9

MAIN:
    ; Leer botones
    in temp, PINB

    sbic PINB, 3      ; B3 presionado ; Salta 2 líneas si
PINB[3]=0; Pero si PINB[3]=1, salta una linea
    rjmp SAVE_NUM1

    sbic PINB, 2      ; B2 presionado; Slata 2 líneas si
PINB[2]=0 (switch B3 no presionado); Saltar una linea si
PINB[2]=1 (switch 3 presionado)
    rjmp SAVE_NUM2

    sbic PINB, 1      ; B1 presionado
    rjmp DO_SUM

    sbic PINB, 0      ; B0 presionado
    rjmp DO_SUB

    rjmp MAIN

; --- Guardar Número 1 (R19) y mostrar en LEDs L3-L0 ---
SAVE_NUM1:
    in input, PINC          ; Leer switches en PC3-PC0
    andi input, 0x0F         ; Enmascarar para obtener
solo 4 bits
    mov num1, input          ; Guardar en num1

    ; Limpiar solo bits 3-0 del PORTD (mantener bits 7-4
intactos)
    in temp, PIND
    andi temp, 0xF0
    or temp, num1            ; Insertar num1 en bits 3-0
    out PORTD, temp

    rjmp MAIN

; --- Guardar Número 2 (R20) y mostrar en LEDs L7-L4 ---
SAVE_NUM2:
    in input, PINC          ; Leer switches en PC3-PC0
    andi input, 0x0F
    mov num2, input

    ; Limpiar solo bits 7-4 del PORTD (mantener bits 3-0
intactos)
    in temp, PIND
    andi temp, 0x0F          ; F0 = 11110000 ? OF =
00001111 (mantiene LSB, limpia MSB)
    swap input                ; Mover los bits al nibble
alto
    or temp, input             ; Insertar num2 en bits 7-4
    out PORTD, temp
```

```
rjmp MAIN

; --- Suma de R19 + R20 ---
DO_SUM:
    mov temp, num1
    add temp, num2 ;temp = num1+num2
    mov result, temp
    ;Limpiar leds 7:0 antes de poner el resultado
    ldi temp, 0x00
    out PORTD, temp
    ;Mostrar el reusltado de la suma
    out PORTD, result

    ;Limpiar los prevención los Leds10 y Led9
    in temp, PORTC ; Leer el valor actual de PORTC
    andi temp, 0b11001111 ; Borrar bits 5 y 4 (conservar
    los demás)
    out PORTC, temp ; Escribir el nuevo valor

rjmp MAIN

; --- Resta de R19 - R20 ---
DO_SUB:
    mov temp, num1
    sub temp, num2 ;temp = num1 - num2
    mov diff, temp
    ;Limpiar leds 7:0 antes de poner el resultado
    ldi temp, 0x00
    out PORTD, temp
    ;Mostrar resultado
    out PORTD, diff

    ; Mostrar indicadores LED9 (negativo), LED10 (cero)

    brmi NEGATIVE
    breq ZERO
    rjmp MAIN

NEGATIVE:
    sbi PORTC, 4 ; LED9 prendido con PC4
    rjmp MAIN

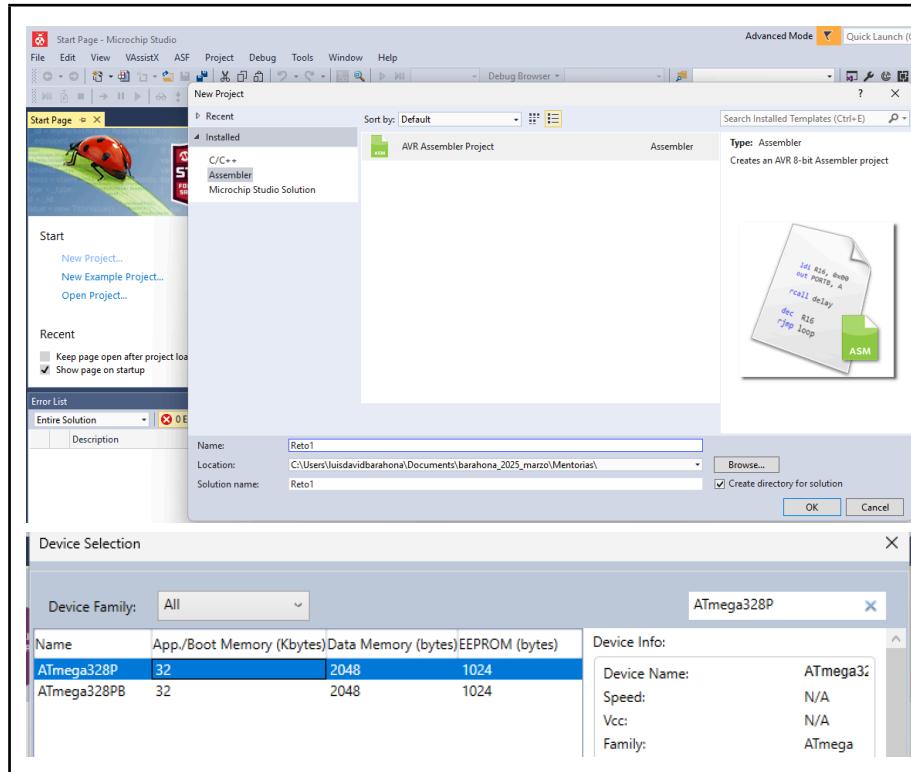
ZERO:
    sbi PORTC, 5 ; LED10 prendido con PC5
    rjmp MAIN
```

2.5. Código en C

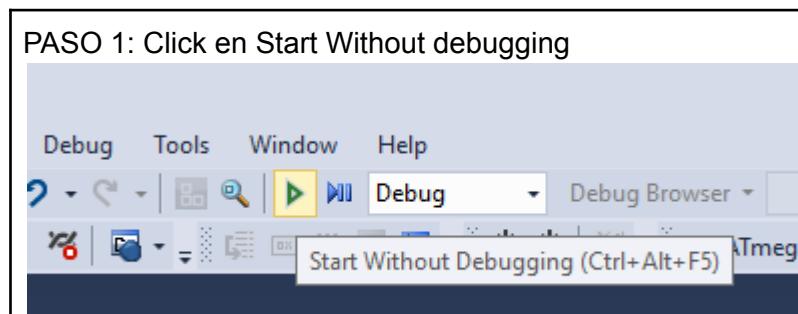
3. Simular la propuesta de solución (etapa de simulación)

3.1. Simulación en Microchip Studio

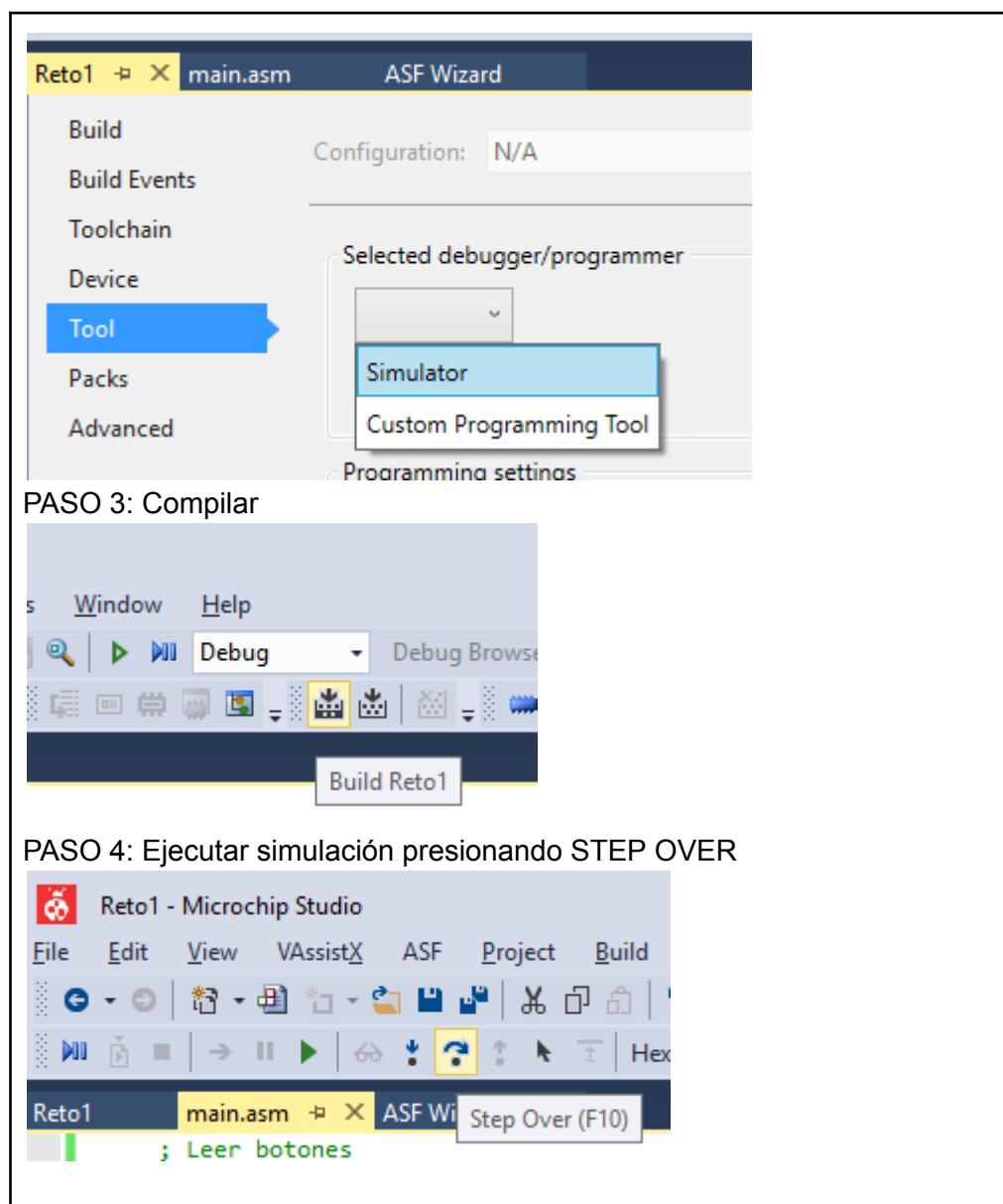
3.1.1. Crear proyecto en Microchip



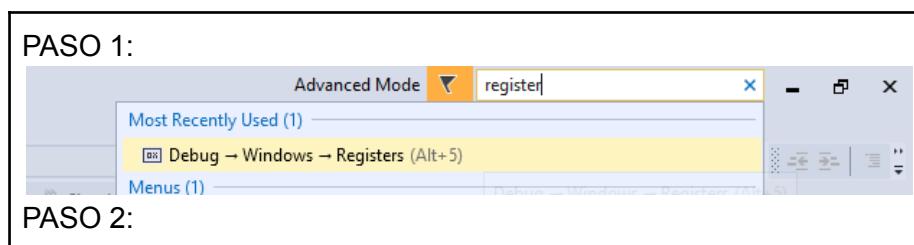
3.1.2. Iniciar simulación

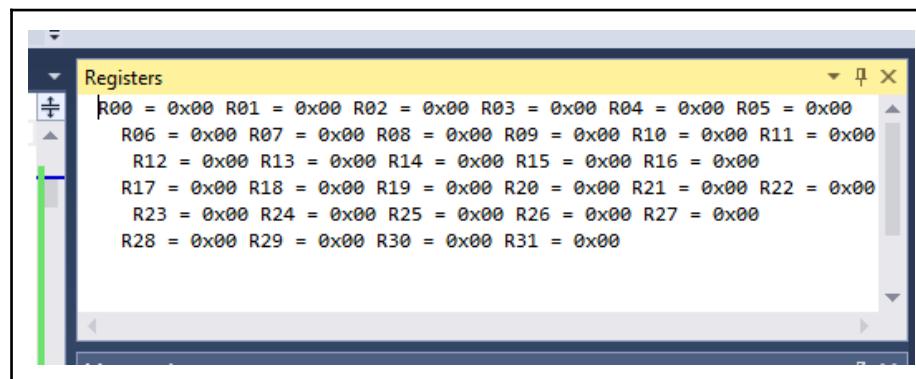


PASO 2: Seleccionar simulador

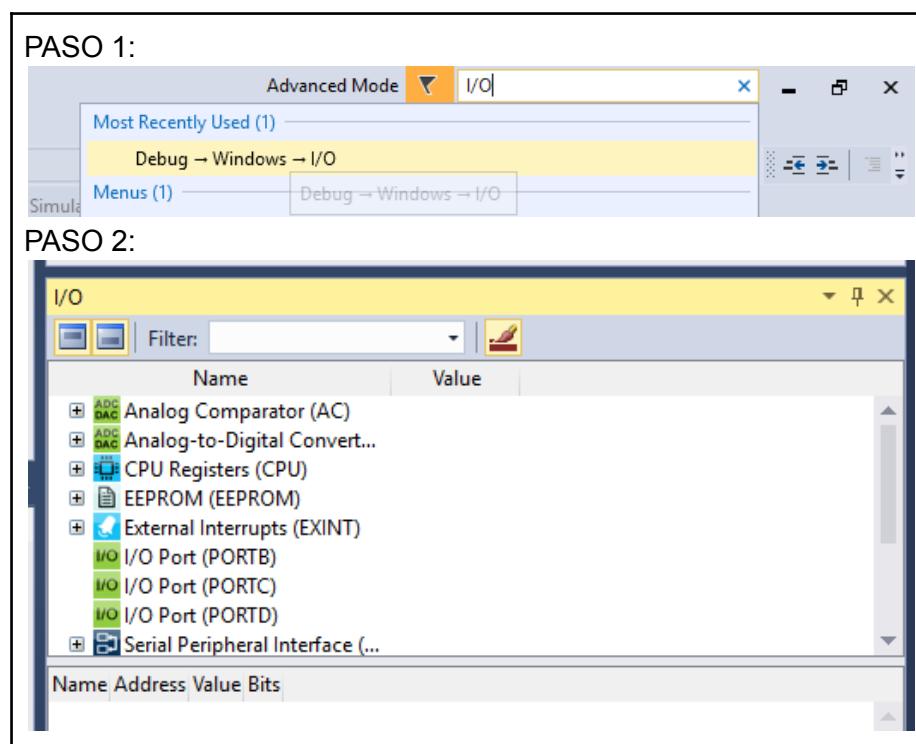


3.1.3. Análisis de registros

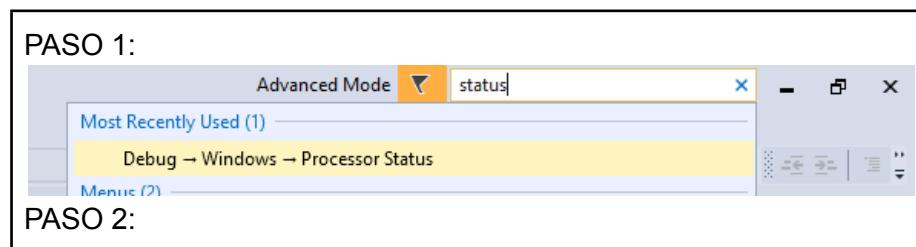




3.1.4. Análisis de puertos I/O



3.1.5. Análisis del registro de banderas o STATUS REGISTER



Name	Value
Program Counter	0x00000000
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	0 0 H S V N Z C
Cycle Counter	0
Frequency	1.000 MHz
Stop Watch	0.00 µs
Registers	
R00	0x00
R01	0x00

3.1.6. Validación de casos

3.1.6.1. Evidencia de configuración adecuada de puertos I/O

SWITCHES DE ENTRADA: SW3:SW0

Name	Address	Value	Bits
I/O PINC	0x26	0x00	0 0 0 0 0 0 0 0
I/O DDRC	0x27	0x00	0 0 0 0 0 0 0 0
I/O PORTC	0x28	0x00	0 0 0 0 0 0 0 0

PUERTOS DE ENTRADA : Pulsador B3:B0

+ External Interrupts (EXINT)			
+ I/O Port (PORTB)			
+ I/O Port (PORTC)			
+ I/O Port (PORTD)			
+ Serial Peripheral Interface (...)			
Name	Address	Value	Bits
I/O PINB	0x23	0x00	0 0 0 0 0 0 0 0
I/O DDRB	0x24	0x00	0 0 0 0 0 0 0 0
I/O PORTB	0x25	0x00	0 0 0 0 0 0 0 0

PUERTOS DE SALIDA DE BANDERAS: Led10 y Led9

I/O I/O Port (PORTC)					
Name	Address	Value	Bits		
I/O PINC	0x26	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRC	0x27	0x30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I/O PORTC	0x28	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

PUERTOS DE SALIDA DE DATOS: Led7:Led0

I/O I/O Port (PORTD)					
Name	Address	Value	Bits		
I/O PIND	0x29	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRD	0x2A	0xFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
I/O POR...	0x2B	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

3.1.6.2. Evidencia de carga del número 1

PASO 1: Seteamos a PINB3 como 1, para indicar que se ha presionado el switchr B3

Name	Address	Value	Bits		
I/O PINB	0x23	0x08	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I/O DDRB	0x24	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I/O PORTB	0x25	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

PASO 2: Seteamos número 1 en 5 (0101) como ejemplo

I/O I/O Port (PORTC)					
Name	Address	Value	Bits		
I/O PINC	0x26	0x05	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I/O DDRC	0x27	0x30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I/O PORTC	0x28	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

PASO 3: Verificar que el número 1 se muestre en el LSB del puerto D

I/O I/O Port (PORTD)				
Serial Peripheral Interface (...)				
Name	Address	Value	Bits	
I/O PIND	0x29	0x05	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRD	0x2A	0xFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
I/O PORTD	0x2B	0x05	<input type="checkbox"/>	<input type="checkbox"/>

3.1.6.3. Evidencia de carga del número 2

PASO 1: Seteamos a PINB2 como 1, para indicar que se ha presionado el switchr B2. Además reseteamos el switchr B3.

Name	Address	Value	Bits	
I/O PINB	0x23	0x04	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRB	0x24	0x00	<input type="checkbox"/>	<input type="checkbox"/>
I/O PORTB	0x25	0x00	<input type="checkbox"/>	<input type="checkbox"/>

PASO 2: Seteamos número 2 en 7 (1111) como ejemplo

Name	Address	Value	Bits	
I/O PINC	0x26	0x07	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRC	0x27	0x30	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I/O PORTC	0x28	0x00	<input type="checkbox"/>	<input type="checkbox"/>

PASO 3: Verificar que el número 2 se muestre en el MSB del puerto D

Serial Peripheral Interface (...)				
Name	Address	Value	Bits	
I/O PIND	0x29	0x05	<input type="checkbox"/>	<input type="checkbox"/>
I/O DDRD	0x2A	0xFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
I/O PORTD	0x2B	0x75	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.1.6.4. Evidencia de la suma del número 1 y número 2

PASO 1: Setear en 1 el switchr B1, para habilitar la suma

Name	Address	Value	Bits	
I/O PINB	0x23	0x02	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I/O DDRB	0x24	0x00	<input type="checkbox"/>	<input type="checkbox"/>
I/O PORTB	0x25	0x00	<input type="checkbox"/>	<input type="checkbox"/>

PASO 2: Visualizar resultado en el puerto D

Name	Address	Value	Bits
I/O PIND	0x29	0x00	□□□□□□□□□
I/O DDRD	0x2A	0xFF	■■■■■■■■■■■■■■■■■■
I/O PORTD	0x2B	0x0C	□□□□□■■■□□□□

3.1.6.5. Evidencia de la resta del número 1 número 2

PASO 1: Setear en 1 el switchr B0 para habilitar la resta

Name	Address	Value	Bits
I/O PINB	0x23	0x01	□□□□□□□□■
I/O DDRB	0x24	0x00	□□□□□□□□□□
I/O PORTB	0x25	0x00	□□□□□□□□□□

PASO 2: Visualizar el resultado en el puerto D

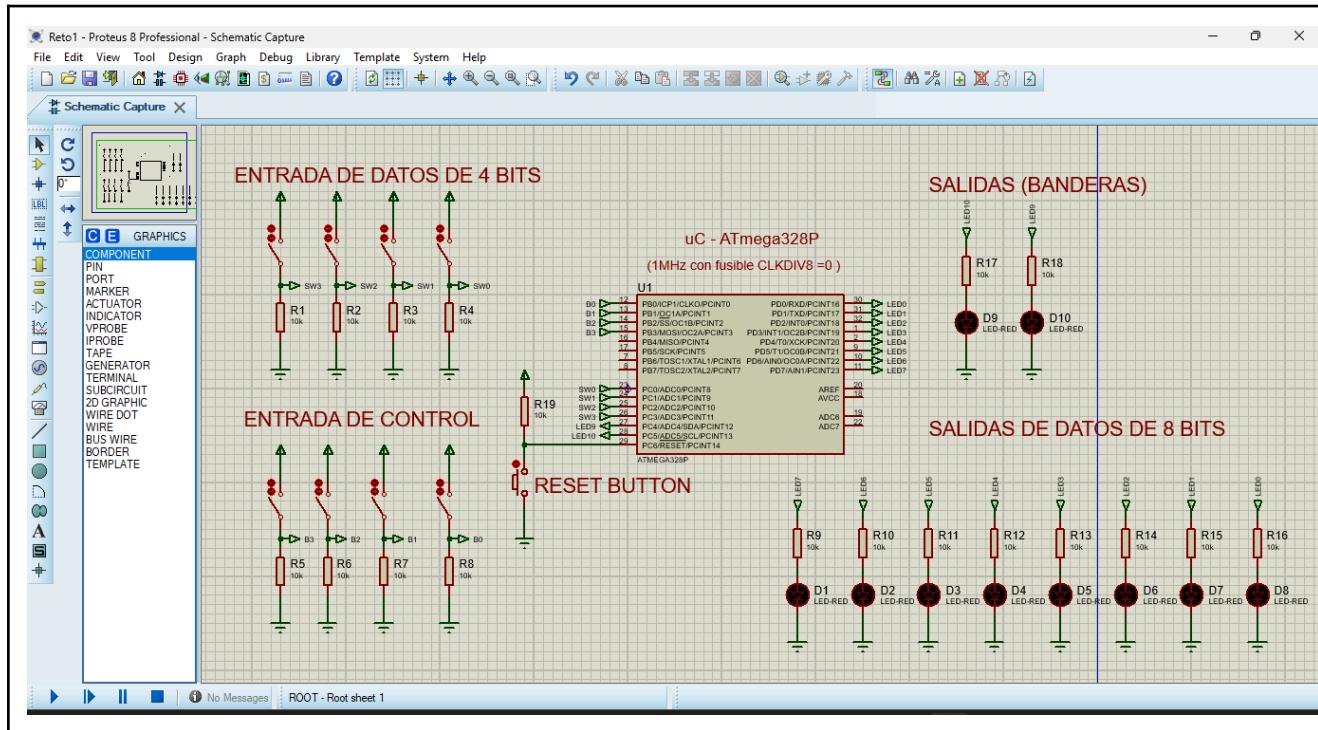
Serial Chip Serial Interface ...

Name	Address	Value	Bits
I/O PIND	0x29	0x75	□■■■■■■□■■■■■■
I/O DDRD	0x2A	0xFF	■■■■■■■■■■■■■■■■■■
I/O PORTD	0x2B	0x00	□□□□□■■■□□□□

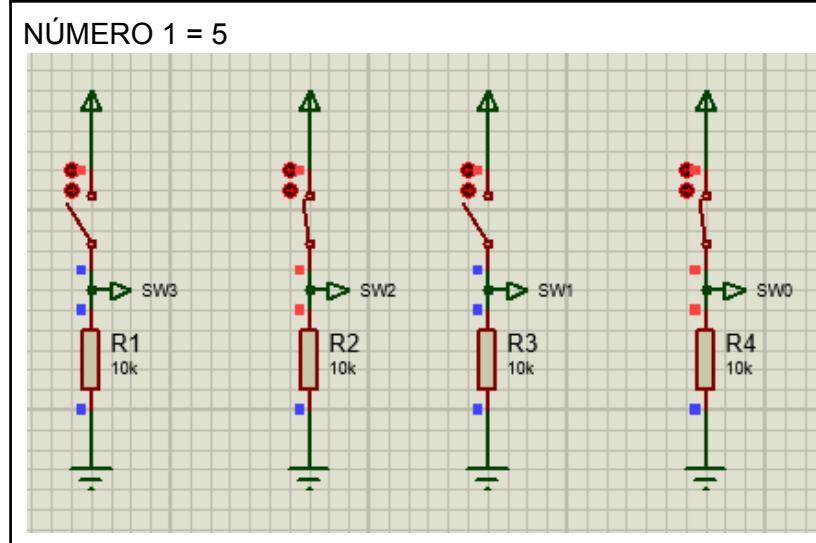
Name	Address	Value	Bits
I/O PIND	0x29	0x00	□□□□□□□□□
I/O DDRD	0x2A	0xFF	■■■■■■■■■■■■■■■■■■
I/O PORTD	0x2B	0xFE	■■■■■■■■■■■■■■■■■■

3.2. Simulación en Proteus 8 Professional

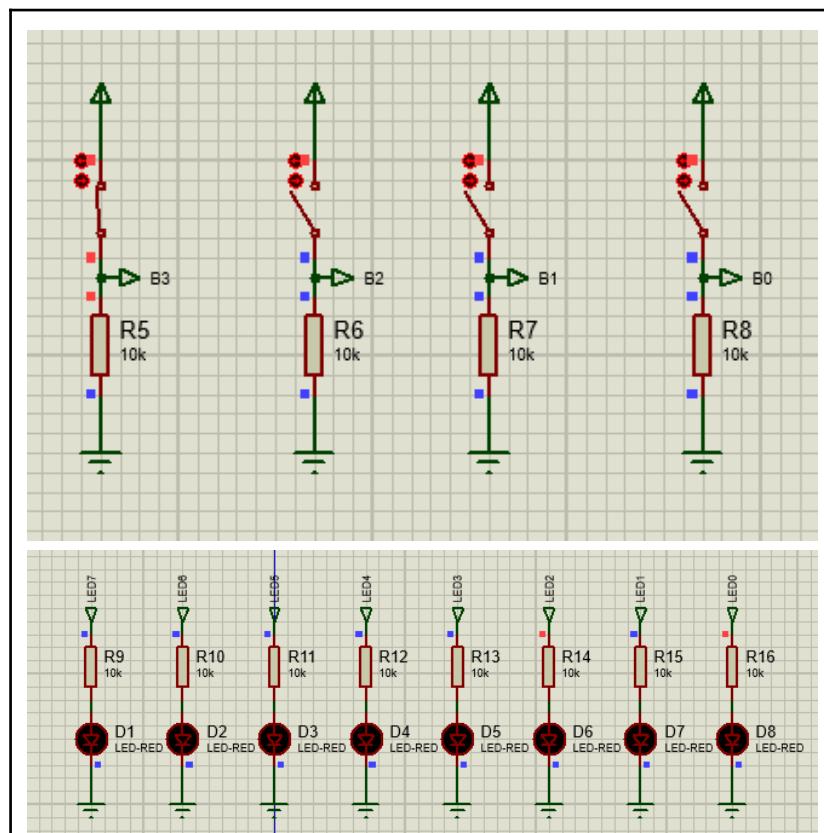
3.2.1. Configuración de ATmega328P sin reloj externo



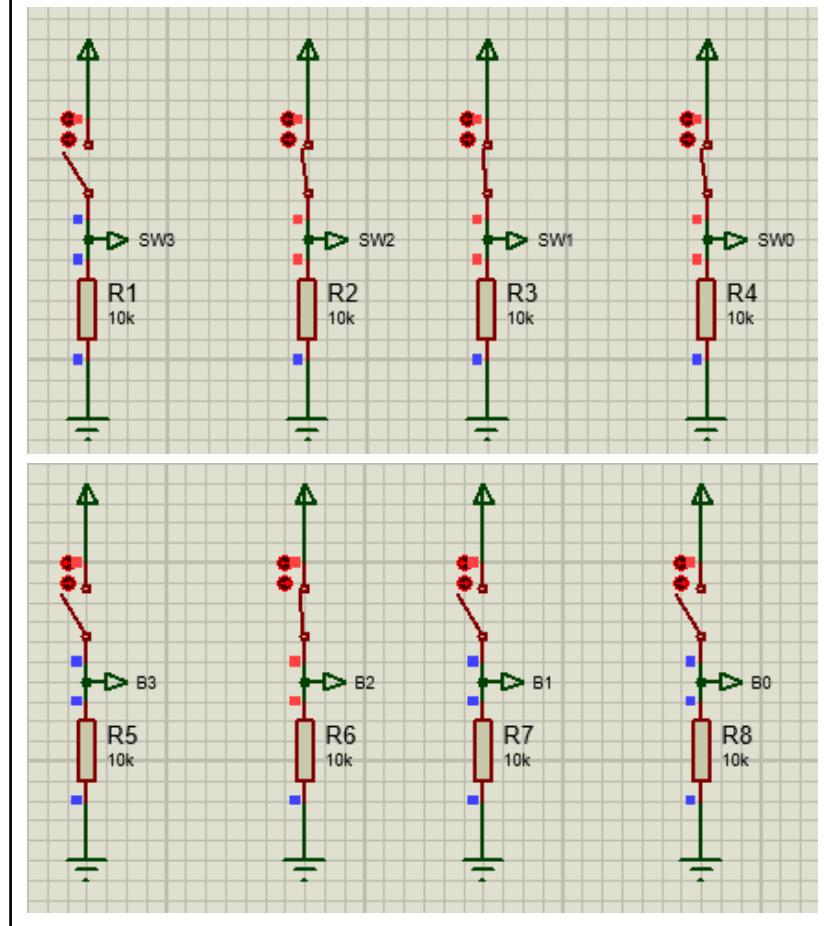
3.2.1.1. Validación de la carga de números

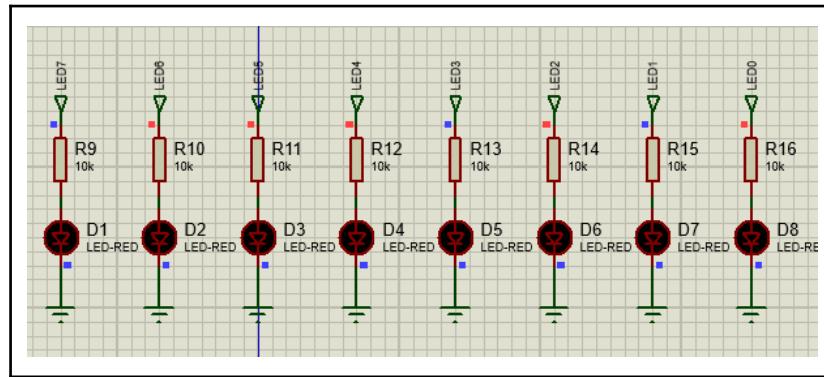


MENTORÍAS ACADÉMICAS 2025-I
INTRODUCCIÓN A SISTEMAS EMBEBIDOS

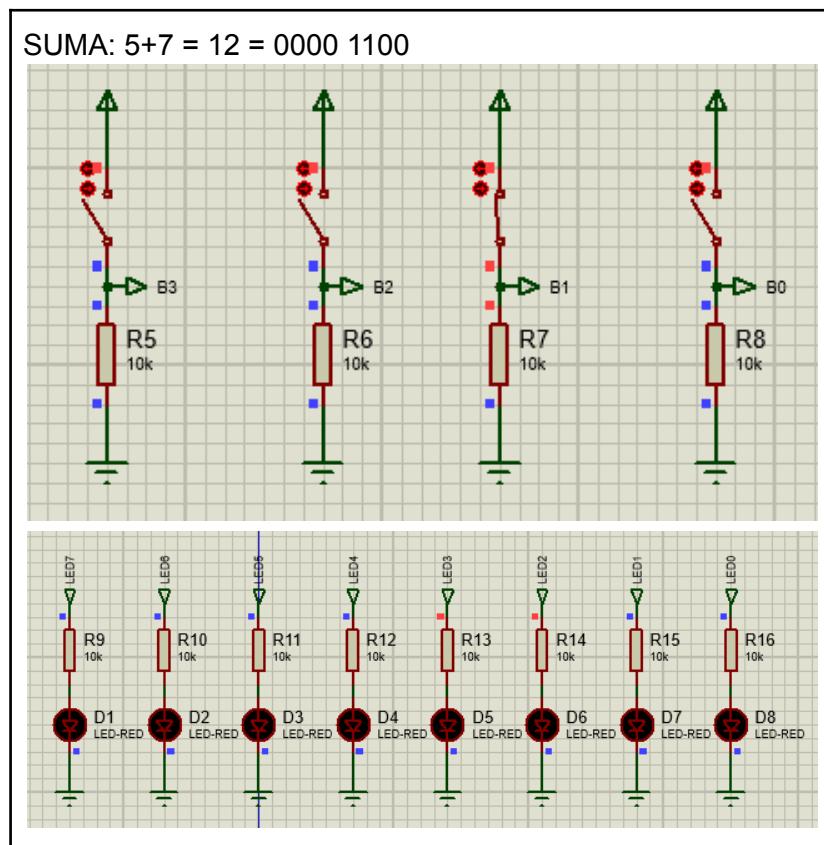


NÚMERO 2 = 7



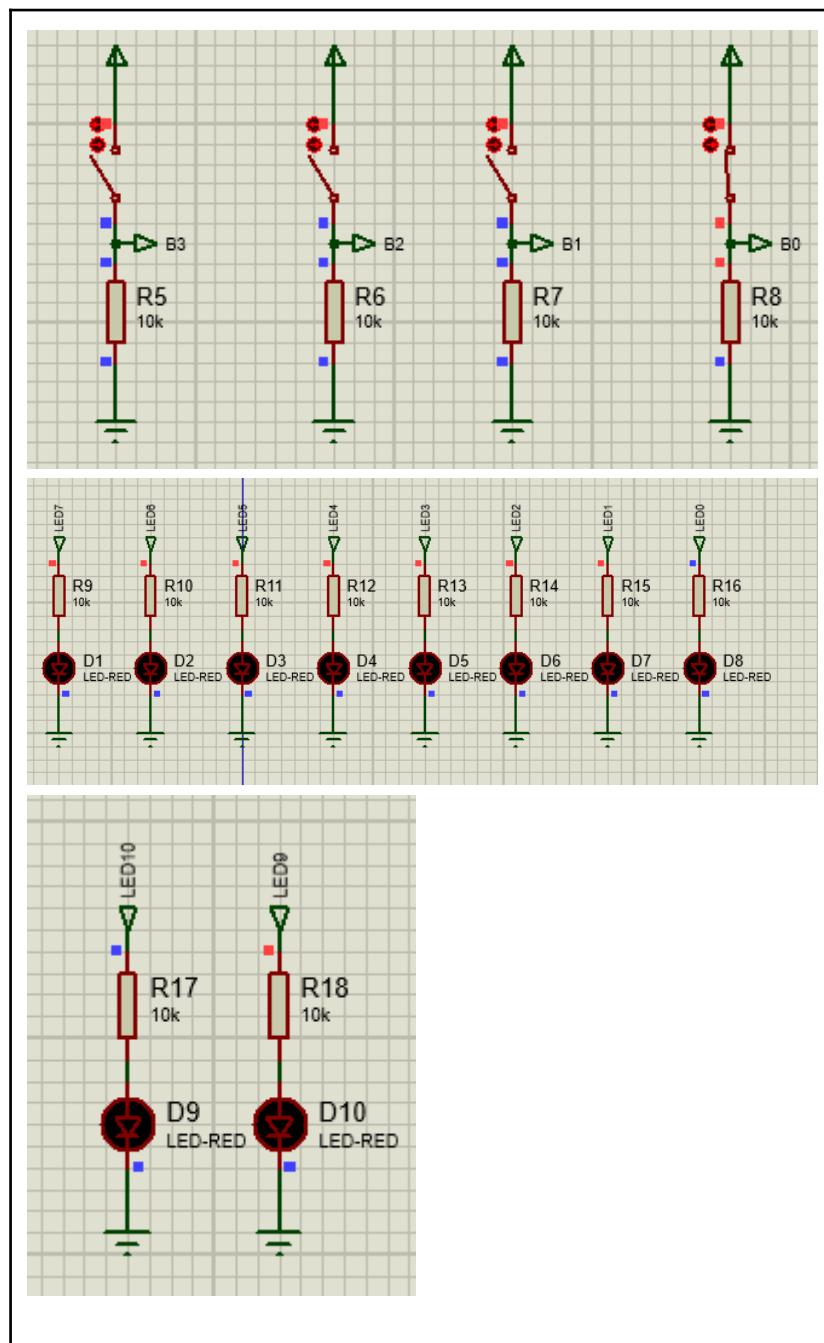


3.2.1.2. Validación de la suma



3.2.1.3. Validación de la resta

RESTA = 5-7=-2 = 1111 1110 (C2)



3.2.2. Configuración de ATmega328P con reloj externo



4. Prototipar la propuesta de solución (etapa de prototipado)

4.1. Diagrama de conexiones

Usar el diagrama creado en proteus

4.2. Validación de casos

5. Implementar la propuesta de solución (etapa de diseño de PCB y fabricación)

- Altium Academy -> Certificado free
- MMJ SMART ELECTRONICS -> curso de diseño de PCBs

5.1. Diseño de componentes

5.2. Diseño de esquemático

5.3. Diseño de PCB

5.4. Generación de BOM

5.5. Generación de archivos de fabricación

6. Documentación en Github

6.1. SSH

7. Anexos

7.1. Páginas web adicionales

- <https://highlight.hohli.com/> : Permite darle un estilo más colorido a tu código.



A screenshot of a web-based code editor titled "highlight.hohli.com". The interface includes tabs for "Language" (set to "ARM Assembly"), "Theme", and "Visual Code". The code area contains the following ARM assembly instructions:

```
rjmp MAIN
; --- Suma de R19 + R20 ---
DO_SUM:
```

7.2. Palabras reservadas dentro de Ensamblador AVR (ATmega328P)

Instrucción	Descripción
.include "m328Pdef.inc"	Incluye el archivo con las definiciones de registros del ATmega328P. Ej:

MENTORÍAS ACADÉMICAS 2025-I
INTRODUCCIÓN A SISTEMAS EMBEBIDOS

	PORTRB, DDRC, TCCR0A, etc.	
.org 0x00	Define la dirección de origen del código (en este caso, vector de reset que arranca en 0x00).	
.def	Define un alias para un registro. Por ejemplo: .def temp = r16 te permite usar temp en lugar de r16 para más legibilidad.	
.equ	Asigna un valor constante a un nombre	.equ LED_MASK = 0xF0
.cseg	Inicia una sección de código (code segment)	.cseg
.dseg	Inicia una sección de datos (data segment)	.dseg
.byte	Reserva espacio para una o más variables de 1 byte en RAM	var1: .byte 1
.word	Reserva 2 bytes	var2: .word 1
.dw	Define una palabra (word) de datos constantes (2 bytes)	.dw 0x1234
.db	Define bytes constantes	.db 0x12, 0x34
.end	Finaliza el archivo de ensamblador	.end
.macro / .endm	Define una macro personalizada	.macro LED_ONendm

7.3. Lista de instrucciones lógicas, aritméticas y de salto usadas en el ejercicio de ejemplo

rjmp	Salto relativo a otra línea	rjmp MAIN
------	-----------------------------	-----------

ldi	Carga un valor inmediato a un registro	ldi r16, 0xFF
in/out	Leer o escribir en puertos de E/S	in r16, PORTB
sbi/cbi	Set/clear bit de un registro de E/S	sbi PORTC, 0
sbic/sbis	Salta si bit está en cero/uno	sbic PINB, 3
mov	Copia un registro a otro	mov r17, r19
add /sub	Suma / resta	add r16, r17
swap	Intercambia los nibles alto y bajo de un registro	swap r18
cpi	Compara un registro con un valor inmediato	cpi r18, 0
brmi	Salta si el resultado anterior fue negativo	brmi NEGATIVE
breq/brne	Salta si igual/distinto	breq ZERO
nop	No hace nada, instrucción de espera	nop
ret	retorna de una subrutina	ret

8. Otros

8.1. Ejemplo de ejercicios de salto condicional

- <https://www.youtube.com/watch?v=S3Xbppl-FoY>

8.2. SD