Circuitos Digitales y Microcontroladores **Facultad de Ingeniería UNLP**	**Trabajo Práctico N°1**	**Fecha de Entrega: 11/04/2022** **Profesor:** **Juárez, José María**
FACULTAD DE INGENIERÍA	**Adrián Daniel Barral 01840/5** **Lautaro La Vecchia 02031/2**	

1. Diseño del circuito

1.1 Problema

Se desea conectar 8 diodos LED de diferentes colores al puerto B del MCU y encenderlos con una corriente de 10mA en cada uno. Además se desea conectar un pulsador a una entrada digital del MCU y detectar cuando el usuario presiona y suelta el pulsador.

1.2 Interpretación

Se realizará un esquema eléctrico de la conexión en Proteus, calculando las resistencias en serie necesarias para cada LED, con el fin de asegurar la corriente requerida de 10mA, teniendo en cuenta la caída de tensión particular de cada LED según su color. Se deberá considerar los valores máximos absolutos del microcontrolador a fin de no superar la capacidad de corriente tanto de cada salida individual como de todas las salidas en funcionamiento simultáneo. Se deberá determinar el tipo de conexión para el pulsador (activo en alto o activo en bajo) y para los LEDs (ánodo común o cátodo común).

1.3 Resolución

1.3.1 Cálculo de los resistores

Teniendo en cuenta la tensión de funcionamiento de cada LED y utilizando la Ley de Ohm para la corriente dada, se puede despejar la resistencia requerida en cada caso. Se considera una tensión de alimentación de 5 V.

$V = I * R$

Caída de tensión en cada diodo LED

- *Rojo: 1.8 V*
- *Amarillo: 2.0 V*
- *Azul: 3.0 V*
- *Verde: 2.2 V*

Así entonces, las resistencias serán

- *Para el LED Rojo: 320* Ω
- *Para el LED Amarillo: 300* Ω
- *Para el LED Azul: 200* Ω
- *Para el LED Verde: 280* Ω

Debe notarse que estos valores son producto del cálculo teórico y en la práctica, en caso de no disponer de resistores del valor indicado, deberán reemplazarse por los del valor más próximo superior, a fin de no exceder los 10 mA.

1.3.2 Consideración de valores máximos absolutos

Como la corriente en cada diodo será de 10mA y la corriente individual máxima de cada pin es de 40mA (ver Figura 1), podemos asegurar que se respeta el máximo individual. De manera similar, como la corriente total podría ser de 80 mA (Si cada LED estuviese encendido al mismo tiempo) y la corriente máxima en los pines VCC y GND del MCU es de 200 mA (ver Figura 1), también se respeta este máximo.

28.1 Absolute Maximum Ratings

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	–55		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	–0.5		+13.0	V
Maximum operating voltage		6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA
Injection current at V_{CC} = 0V		±5.0(1)		mA
Injection current at V_{CC} = 5V		±1.0		mA

Note: 1. Maximum current per port = ±30mA

Figura 1. Máximo absolutos. Atmega328p Datasheet.

1.3.3 Esquema de conexión del pulsador y los LEDs

Para la conexión del pulsador, se decidió diseñarlo como un pulsador activo en bajo, para así lograr un diseño más simple al evitar la necesidad de realizar una conexión de alimentación para el mismo. De forma similar, para los LEDs se eligió la conexión de cátodo común, puesto que esto permite utilizar al MCU como fuente, simplificando así el diseño circuital. Esto último, sin embargo, implica como consecuencia que la corriente suministrada a los LEDs no sea exactamente 10mA, ya que el MCU no se comporta como una fuente ideal tensión y al presentar una resistencia interna no podrá otorgar 5V en todo momento a través de sus pines.

2. Diseño del programa

2.1 Problema

Se necesita realizar un programa para que el MCU encienda los LEDs del puerto B con la siguiente secuencia repetitiva:

- b0 y b7 – b1 y b6 – b2 y b5 – b3 y b4 *(Secuencia 1)*

Luego, cuando el usuario presione y suelte el pulsador debe cambiar a la secuencia:

- b3 y b4 – b2 y b5 – b1 y b6 – b0 y b7 *(Secuencia 2)*

Si presiona y suelta nuevamente vuelve a la secuencia original, y así sucesivamente.

2.2 Interpretación

El programa deberá utilizar el puerto B del microcontrolador para encender o apagar cada LED dependiendo de la secuencia en ejecución, con un 1 o 0 respectivamente debido a la conexión de cátodo común elegida. A su vez, el microcontrolador debe leer un pulsador activo en bajo para alternar entre cada secuencia, siendo capaz de detectar tanto la pulsación como el momento en el que el botón es soltado.

Se deberá considerar el tiempo mínimo de parpadeo para que los LEDs sean visibles, y se deberá implementar un mecanismo para evitar el efecto "rebote" de los pulsadores mecánicos.

2.3 Resolución

2.3.1 Setup del programa

Para poder utilizar el puerto B como salida, lo configuramos acordemente colocando la cadena 0xFF en el registro DDRB (un bit en 1 en el registro de direccionamiento indica salida).

Para leer el pulsador, utilizaremos el pin 0 del puerto C (pin C0), configurando el mismo como input, colocando un 0 en la posición correspondiente del registro DDRC. Además, ya que el pulsador es activo en bajo, será necesario configurar en el pin en modo de alta impedancia, es decir utilizando un *resistor pull up*, colocando un 1 en la posición 0 del registro PORTC.

2.3.2 Lectura del pulsador

Para poder reconocer cuando el pulsador se oprime y se suelta se utilizará una variable booleana (**presionado**) junto con el valor del pin C0; esta variable tomará el valor verdadero mientras se mantenga presionado el pulsador, falso en caso contrario.

De forma iterativa se consulta si el botón está siendo presionado. En el momento en el que se detecta este evento, se modifica el estado de la variable **presionado** como si se tratará de una *petición* para el cambio de secuencia, sin embargo, la petición de cambio de secuencia no se llevará a cabo hasta que el botón sea soltado. Diseñar el algoritmo de esta forma permite definir el comportamiento del microcontrolador frente al evento de que el botón se mantenga pulsado: la secuencia actual se seguirá ejecutando hasta que se suelte, momento en el cual se comenzará a ejecutar la nueva secuencia. La selección de secuencia se almacena en una variable booleana, siendo un valor **true** la secuencia 1, y un valor **false** la secuencia 2.

2.3.3 Efecto rebote

A fin de evitar el efecto rebote del botón, se implementó una solución sencilla mediante software. La misma consiste en detener la ejecución durante una breve cantidad de tiempo, a fin de permitir que la señal del pulsador se estabilice. Debe notarse que esta solución, si bien simple, implica un mal aprovechamiento del tiempo de ejecución, y afecta la performance, por lo tanto se debe tener cuidado con el tiempo de retardo elegido ya que un tiempo muy grande puede percibirse como una “falta de respuesta” del dispositivo, y un tiempo muy corto puede resultar en una lectura poco confiable. En este caso se consideró un delay de 50 ms de manera prudencial, considerando el estado transitorio de un pulsador arbitrario promedio.

2.3.4 Secuencia de parpadeo

Se identificó que cada una de las secuencias consiste en encender sucesivamente dos LEDs al mismo tiempo, por un breve momento, desde los extremos hacia el centro o bien desde el centro hacia los extremos.

Para modelar esta situación, puede pensarse a los 8 LEDs como una cadena de 8 bits, con partes alta y baja de 4 bits (Ver Figura 2). Con esa analogía en mente:

- *Secuencia 1:* la parte alta inicia con el bit más significativo encendido y la misma se va desplazando hacia la derecha; la parte baja inicia con el bit menos significativo encendido, y debe desplazarse hacia la izquierda.

- S*ecuencia 2*: la parte alta comienza con el bit menos significativo encendido y se desplaza hacia la izquierda, mientras que la parte baja comienza con el bit más significativo encendido y se desplaza hacia la derecha.

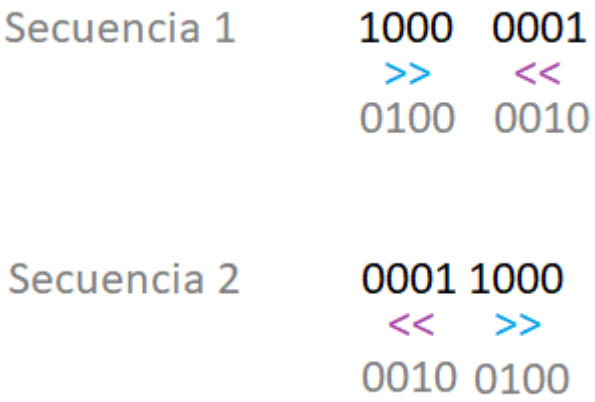


Figura 2. Modelo de la situación.

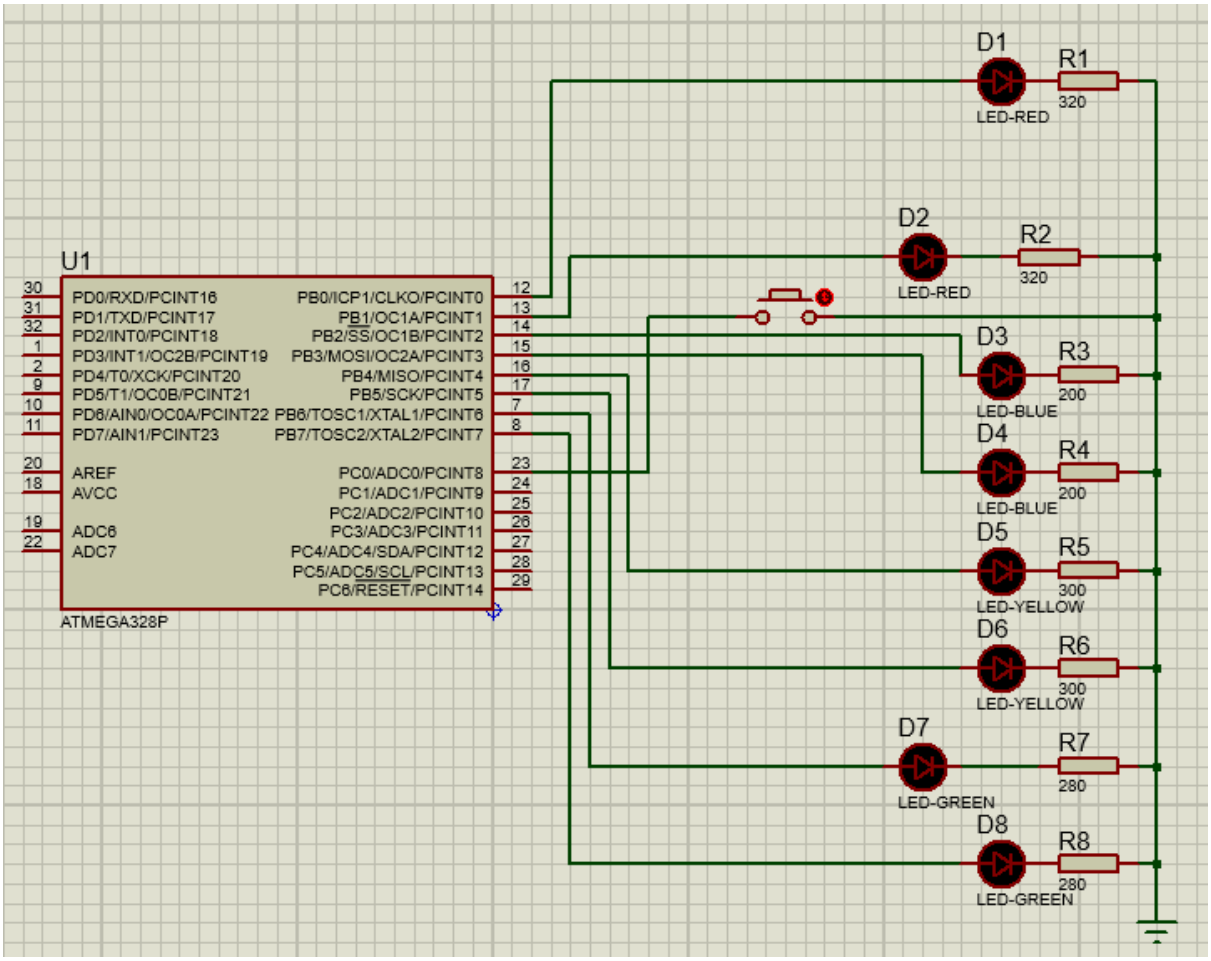
Continuando con la modelización, la secuencia de encendido de LEDs puede pensarse como una serie de 4 etapas, en las que se encienden 2 LEDs cada vez, hasta haberlos encendido todos, momento en el cual se inicia la serie nuevamente. Realizar la secuencia en etapas nos permite tener una mayor frecuencia de consulta al estado del pulsador ya que, contrariamente a ejecutar toda la secuencia en una sola iteración, de esta manera sólo se debe esperar el tiempo que permanecen prendidos 1 par de LEDs para volver a consultar.

Como observación adicional, se decidió configurar una frecuencia de parpadeo de 50 ms para obtener una mayor claridad visual, sin embargo, la misma podría reducirse hasta los 20 ms, la cual determinamos que es la mayor frecuencia de parpadeo distinguible por nuestro ojo.

3. Validación

3.1 Circuito

Se modeló el circuito eléctrico en Proteus para la simulación. El mismo consiste de un pulsador activo en bajo y 8 LEDs conectados en configuración de cátodo común.



3.2 Corriente suministrada a los LEDs

En concordancia con lo mencionado en la sección 1.3.3, se observa que la corriente se encuentra por debajo de los 10 mA. Midiendo la tensión en cada resistor podemos saber que esto se debe a que la tensión entregada por el MCU en los pines es menor a 5V y que varía dependiendo del consumo, comportándose como una fuente real de tensión.

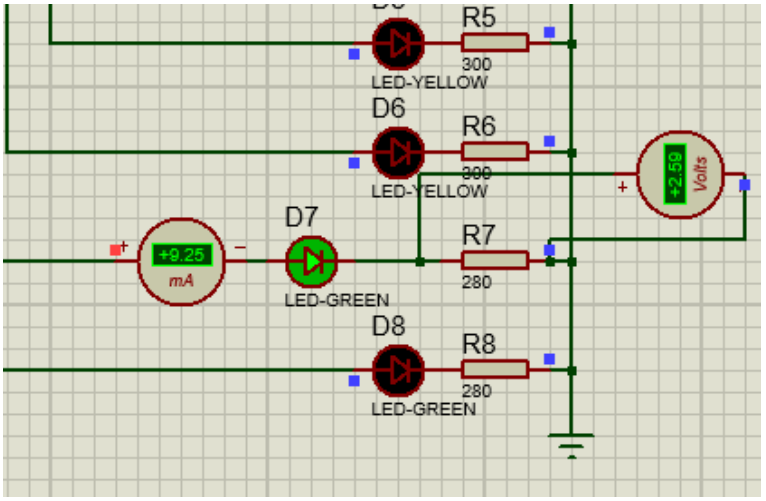


Figura 3. Corriente a través del LED verde.

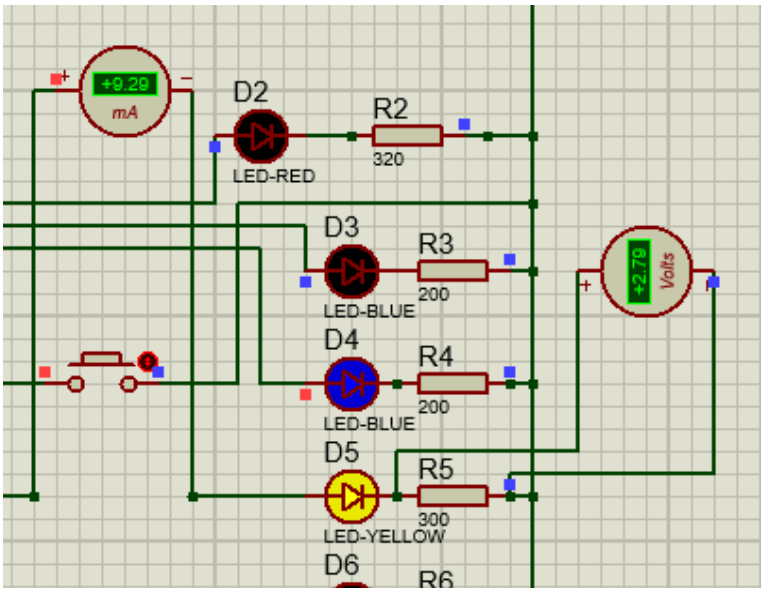


Figura 4. Corriente a través del LED amarillo.

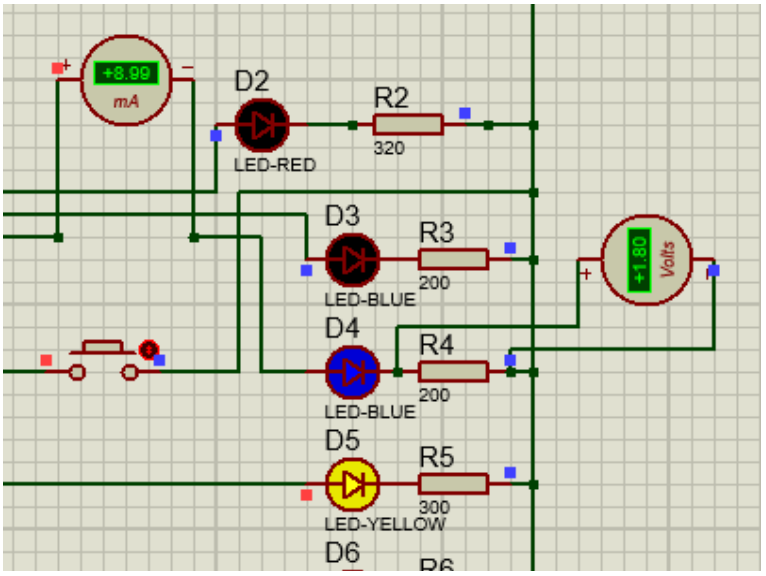


Figura 5. Corriente a través del LED azul.

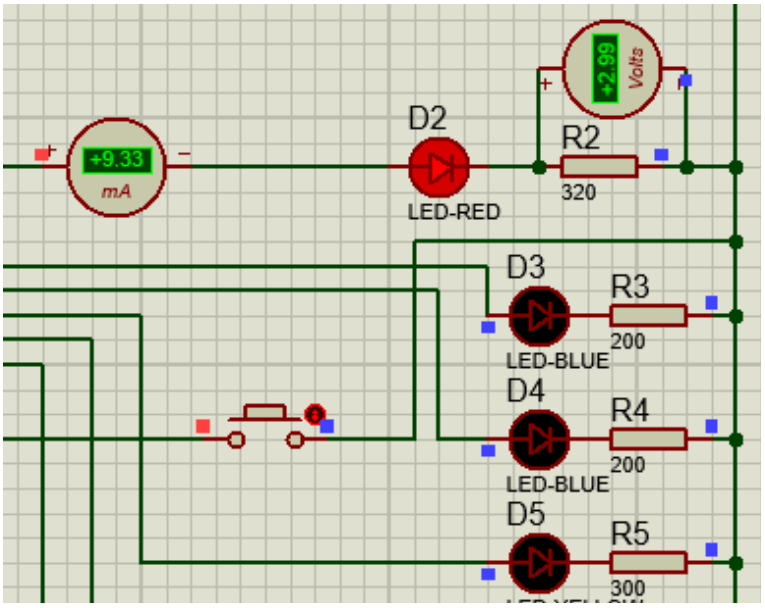
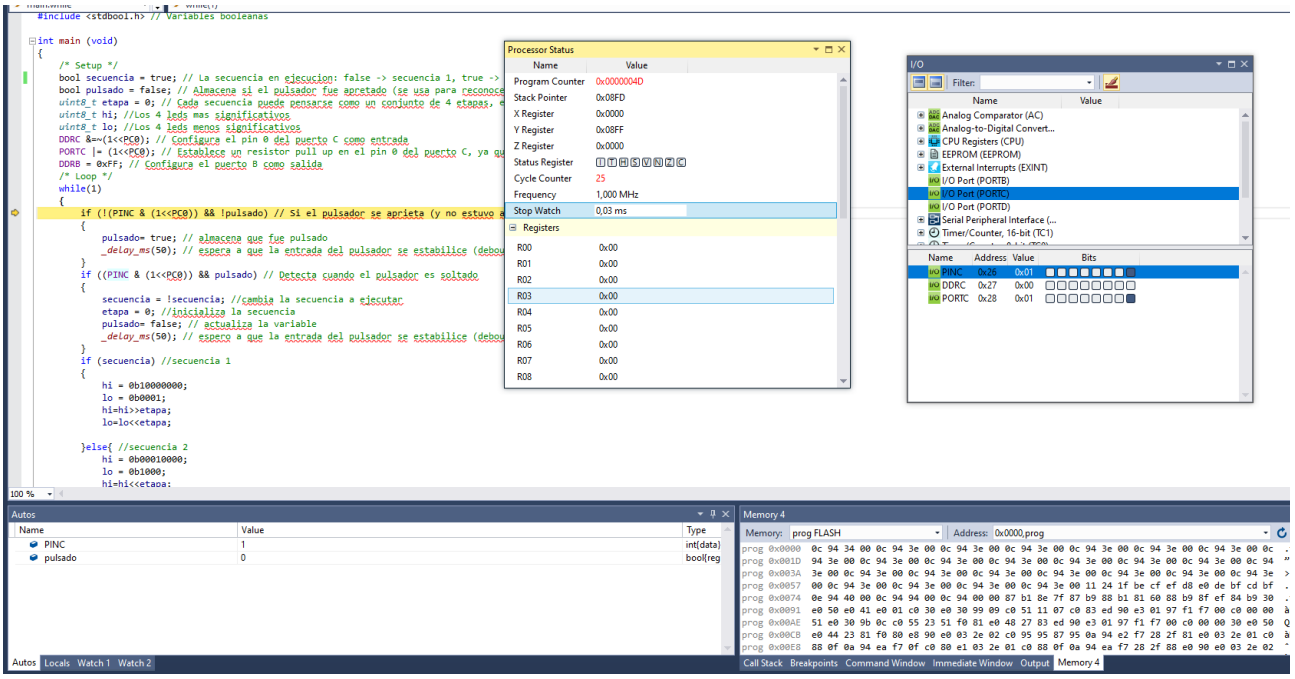


Figura 6. Corriente a través del LED rojo.

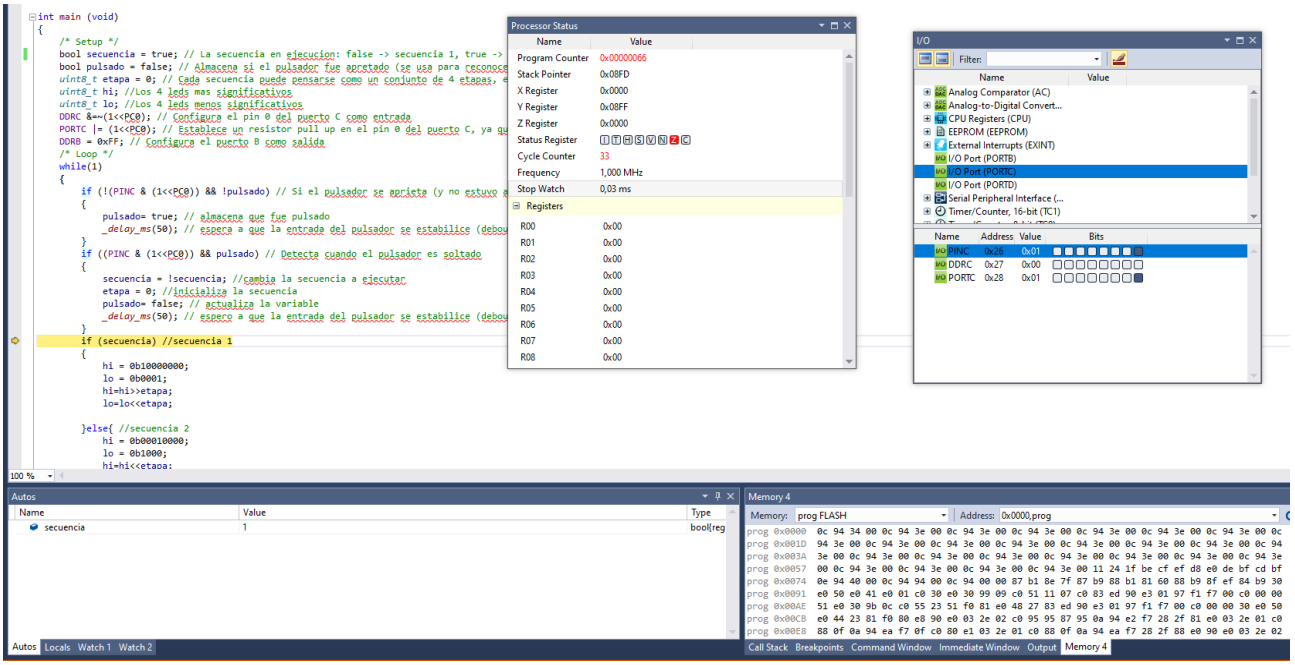
3.3 Simulación

3.3.1 El pulsador no se oprime

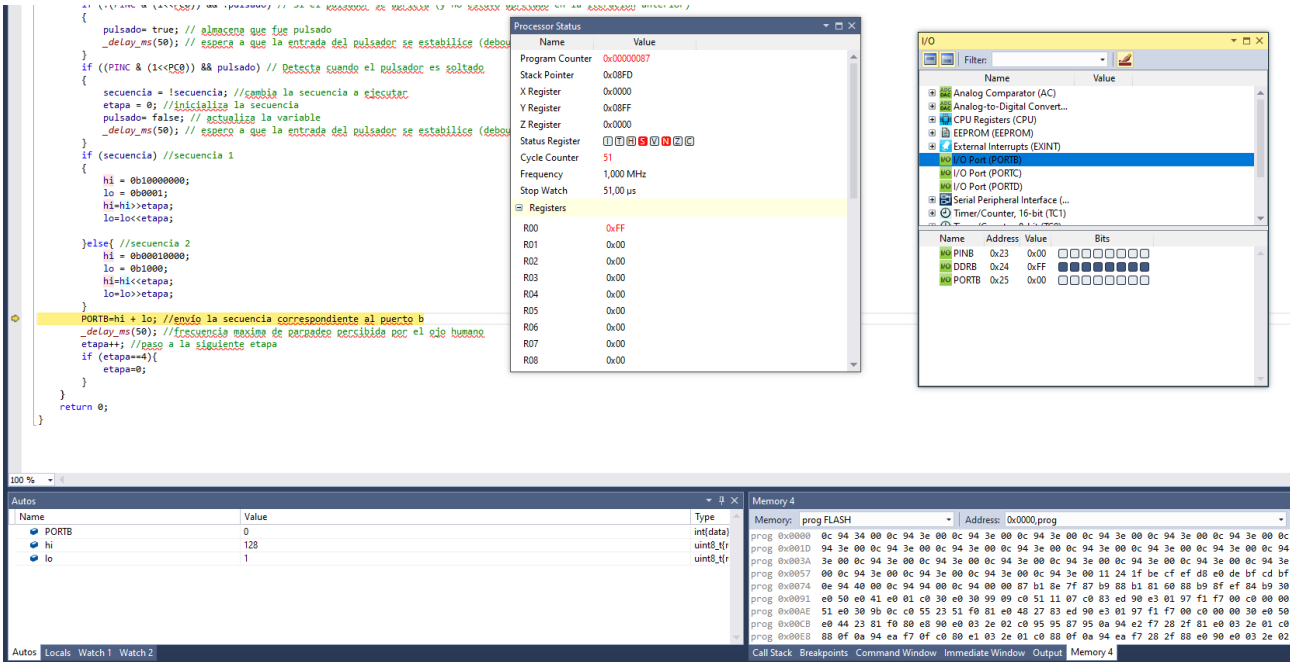
- Duración: cada iteración dura 50 ms. La secuencia completa se ejecuta en 200 ms.
- Observaciones: Si no se oprime el pulsador, el programa no ejecuta ningún bloque if, por lo que solo ejecuta las operaciones relacionadas mostrar a la secuencia de LEDs. Como los LEDs deben permanecer encendidos durante 50 ms, la ejecución se detiene durante ese tiempo.



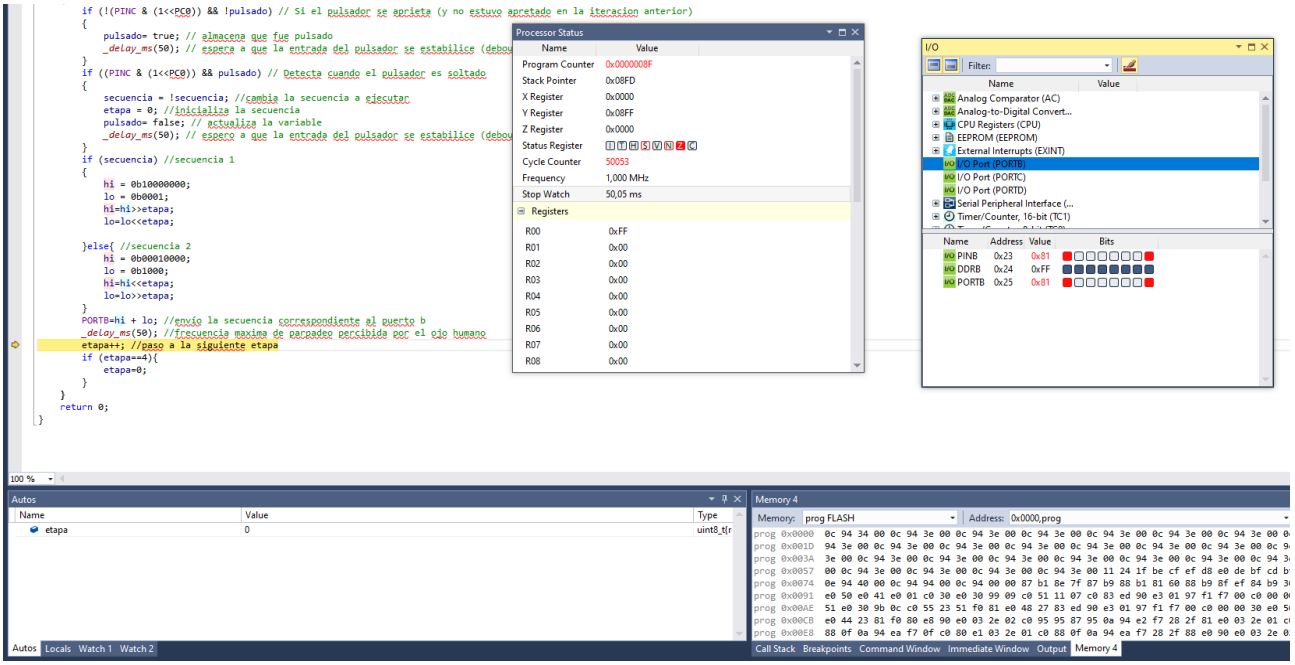
La inicialización toma 0.03 ms aproximadamente. PINC Tiene el valor 0x01 indicando que el pulsador no está activado.



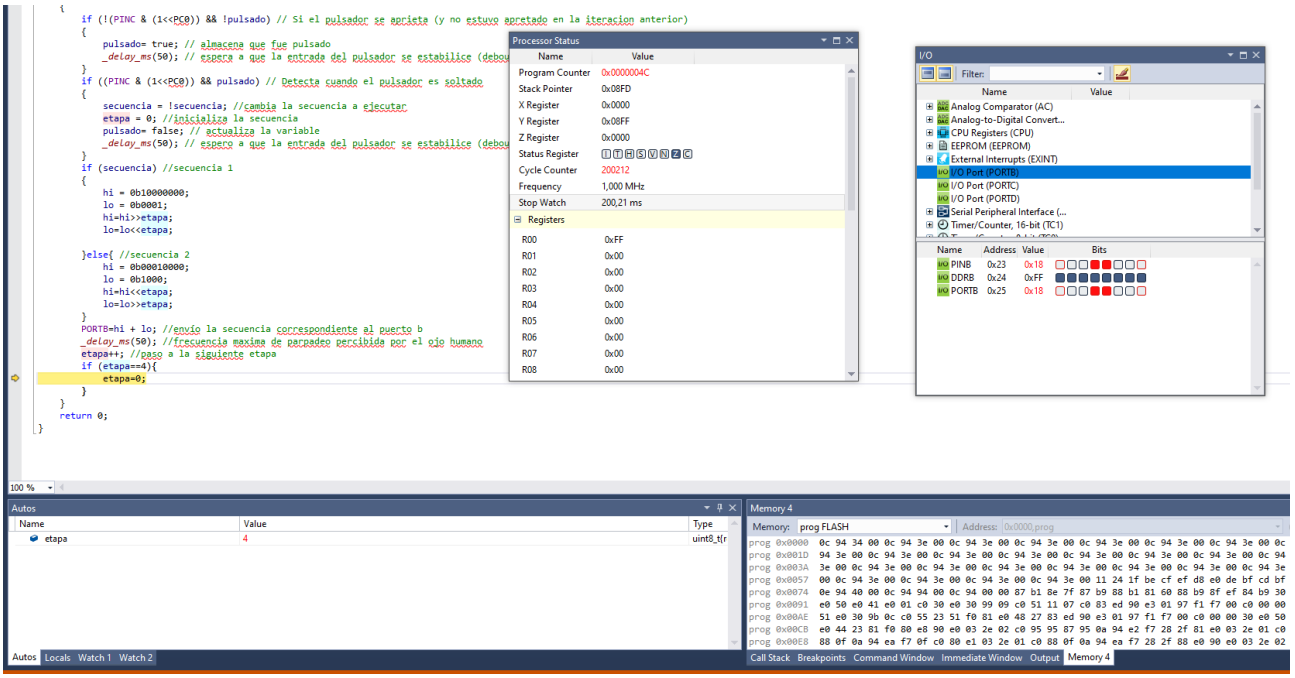
No se ejecutan los if correspondientes al pulsador.



La primera etapa de la secuencia toma 0.02 ms en calcularse y mostrarse.



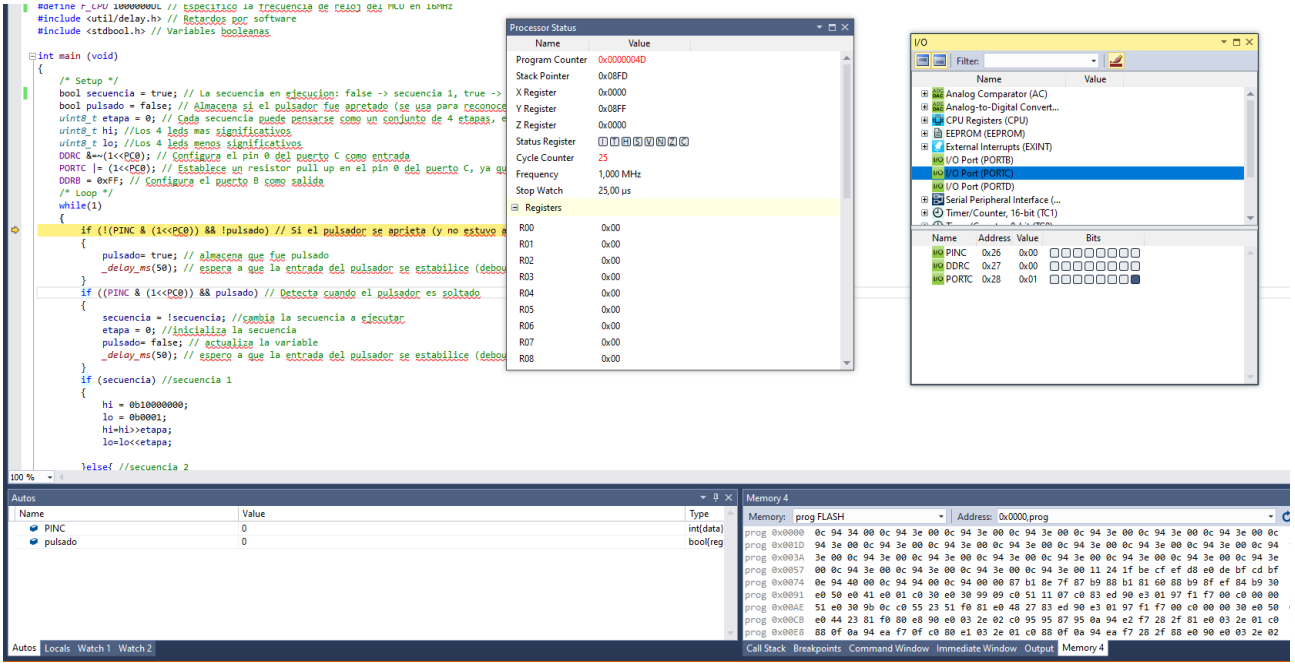
Se muestra por el puerto B la cadena 0x81. Se encienden los bits correspondientes a la primera etapa de la secuencia 1: bit 7 y bit 0. Se mantiene el valor durante 50 ms.



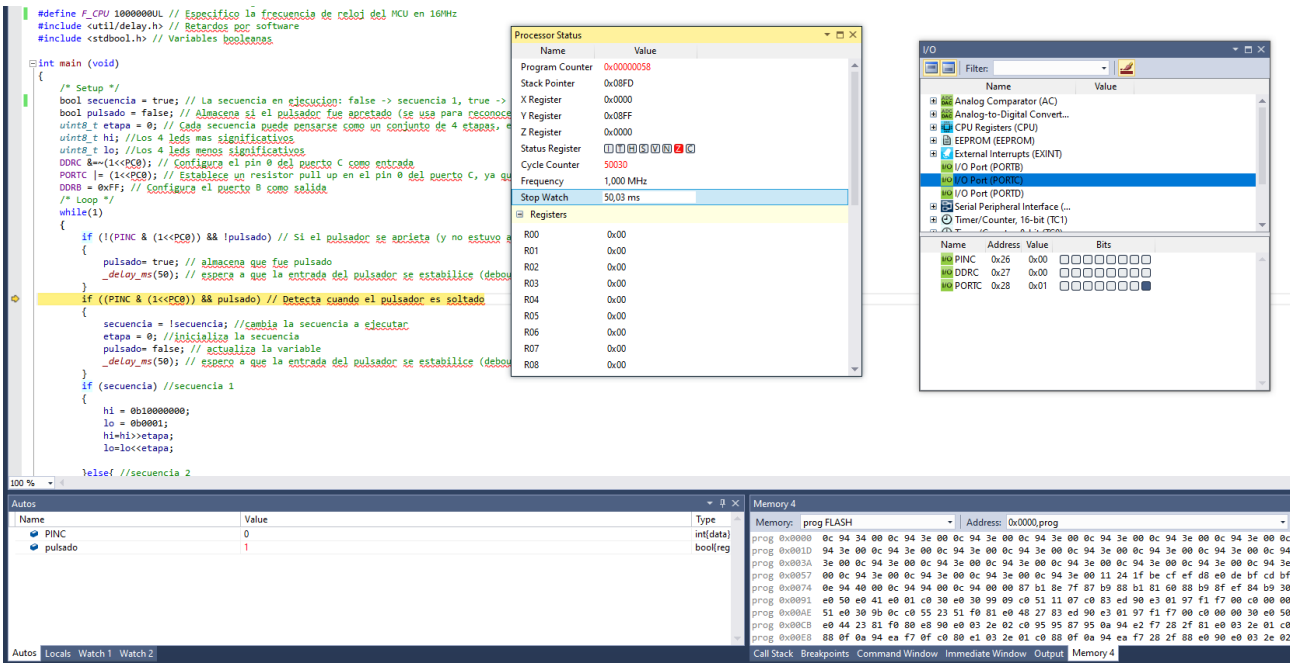
La secuencia tarda aproximadamente 200 ms en ejecutarse, lo esperable teniendo en cuenta que cada etapa dura 50 ms.

3.3.2 El pulsador se mantiene pulsado

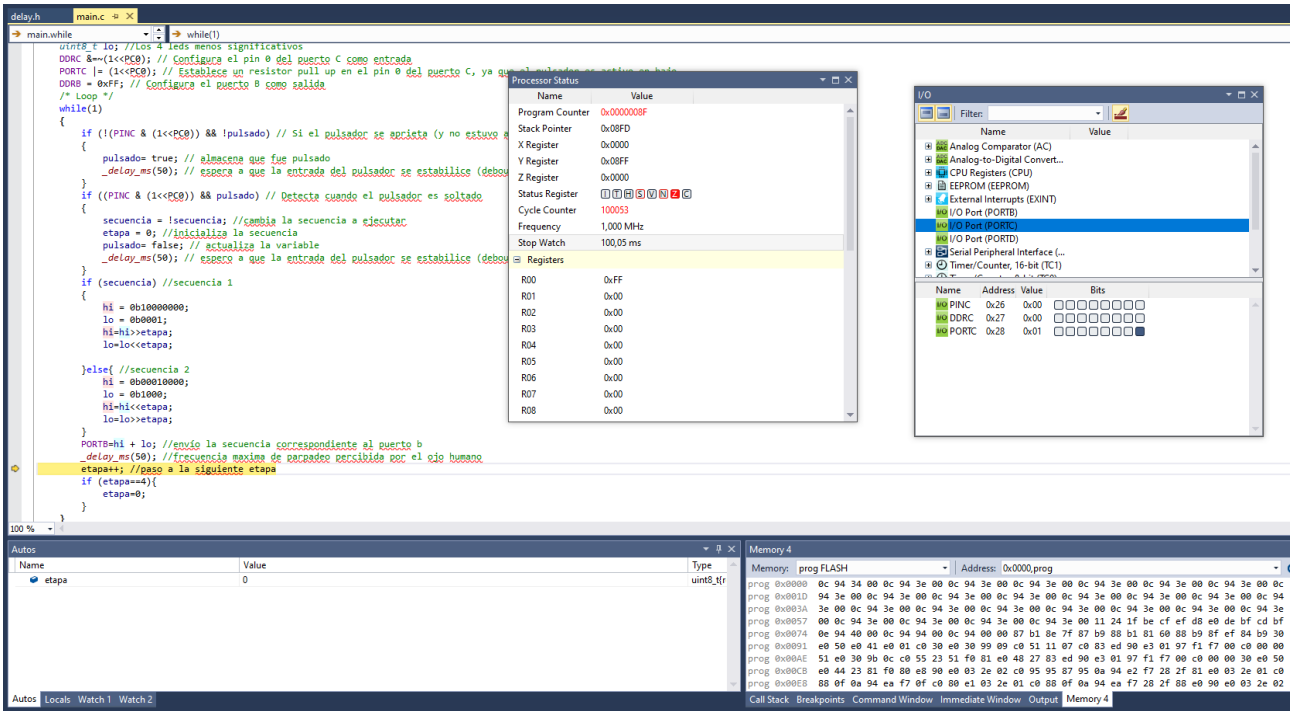
- Duración: La primera iteración dura 100 ms, luego duran 50 ms. La secuencia completa se ejecuta en 250 ms.
- Observaciones: La primera iteración es más larga debido al tiempo que se detiene la ejecución para esperar a que se estabilice la señal del pulsador (*tiempo de debounce*). Al mantenerlo presionado, durante la siguiente iteración el programa reconoce que no hubo ningún cambio en la señal del pulsador, por lo que no hace falta detener la ejecución para esperar que se estabilice la señal. No se ejecuta ningún if y la situación del caso 1 se repite durante el resto de las iteraciones. Como consecuencia la secuencia total tarda en ejecutarse 50 ms adicionales.



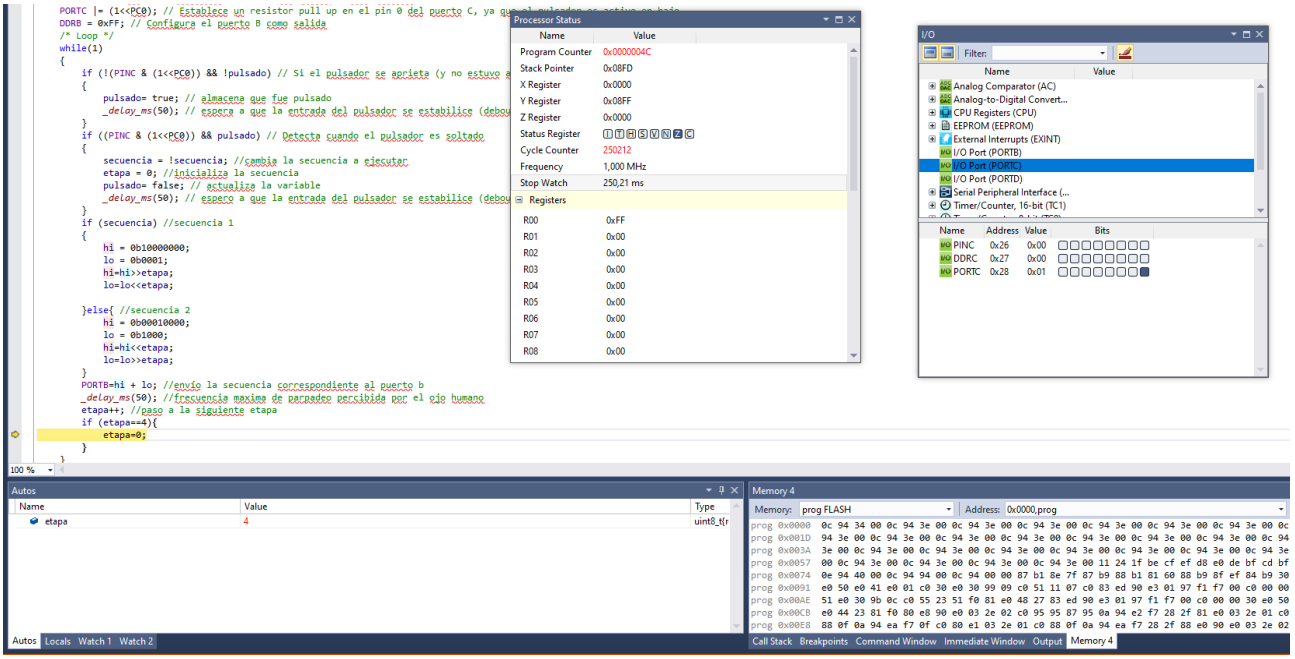
El botón se encuentra presionado, el PINC lee 0x00.



Se ejecuta el primer if, debido al tiempo de debounce se detiene la ejecución durante 50 ms.



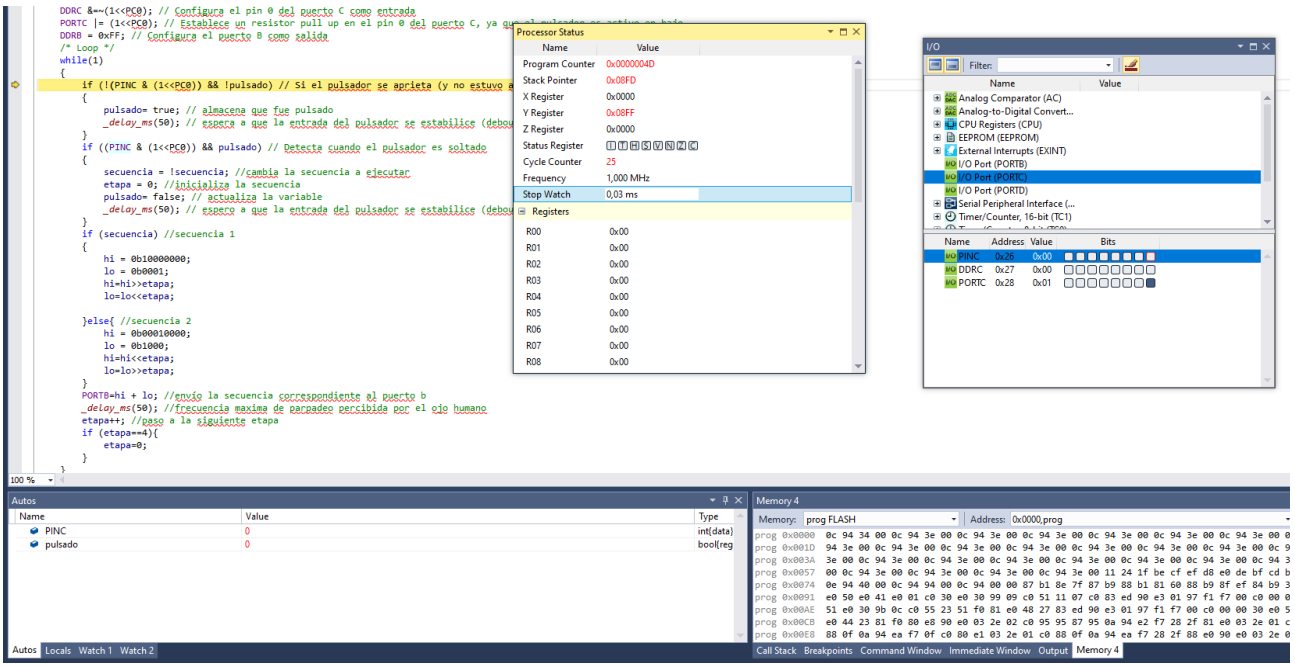
Se envía la secuencia al puerto B y se mantiene el valor durante 50 ms, luego se avanza a la siguiente iteración.



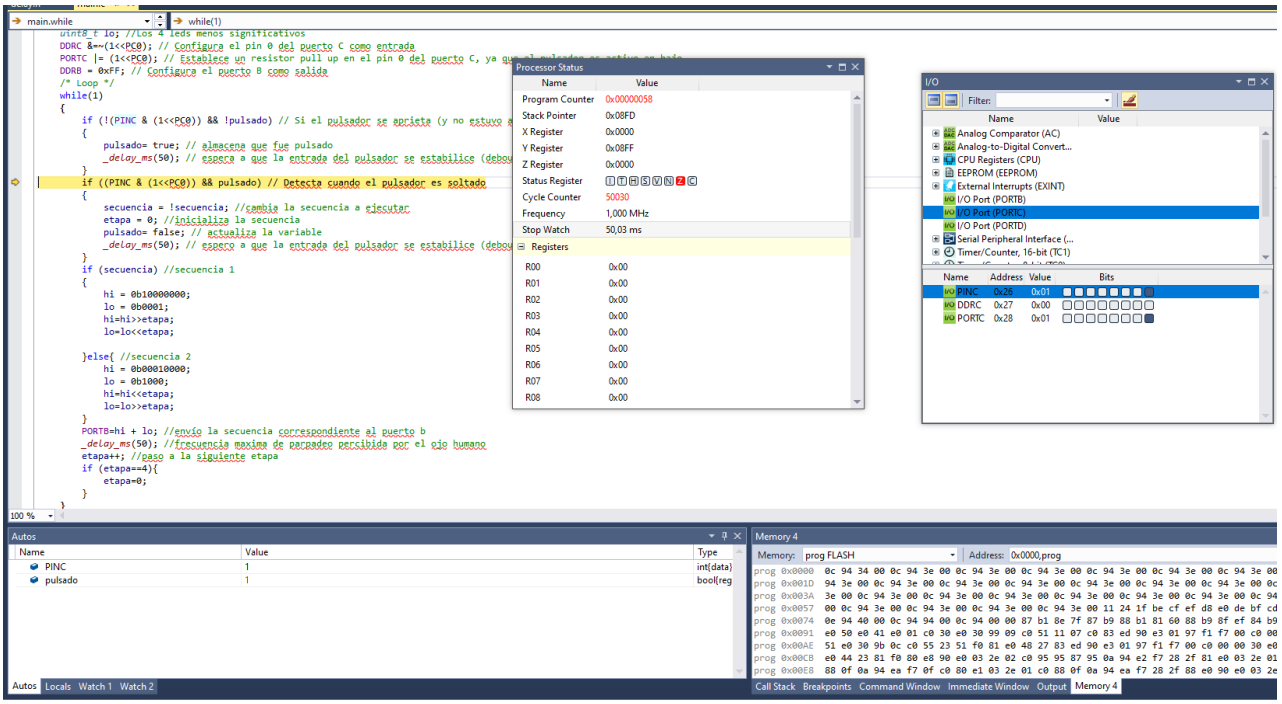
La secuencia tarda 250 ms en completarse, confirmando que a excepción de la primera, las iteraciones duran 50 ms.

3.3.3 El pulsador se oprime y se suelta

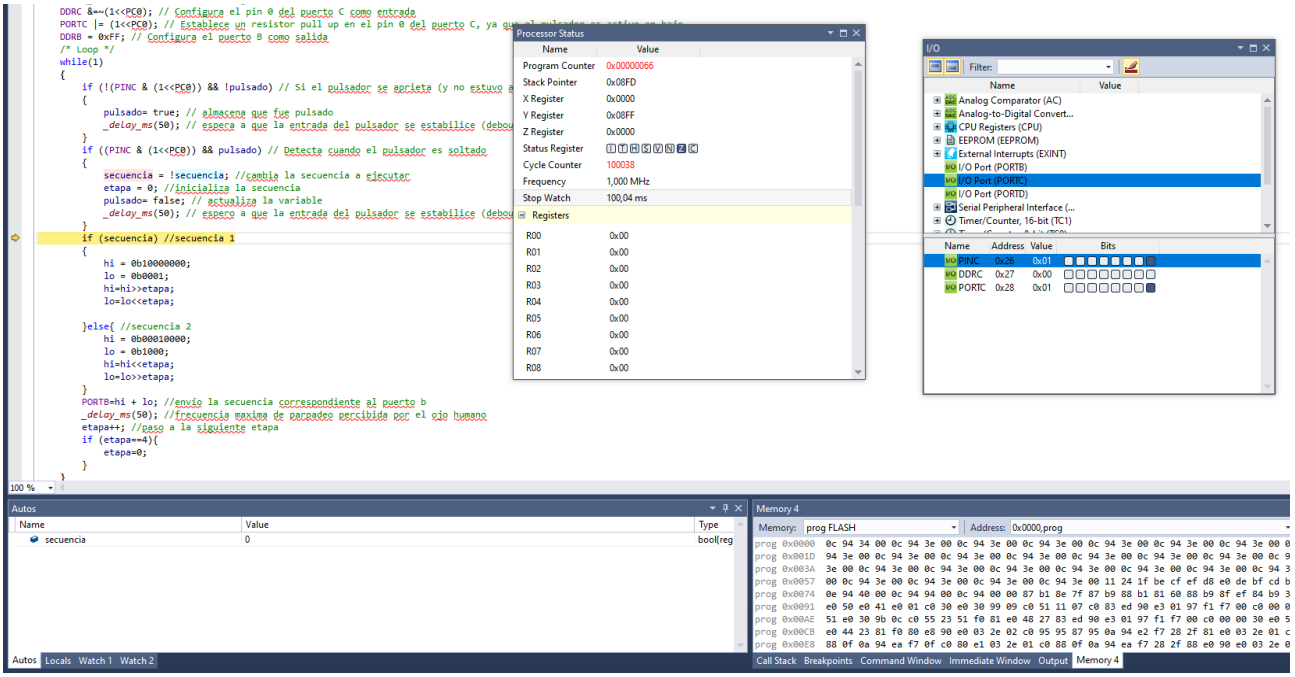
- Duración: La primera iteración tendrá una duración de 150 ms, luego las iteraciones duran 50 ms. La secuencia completa se ejecuta en 300 ms.
- Observaciones: El tiempo de la primera iteración se debe a que debe detenerse la ejecución durante el *tiempo de debounce* en dos ocasiones, para la señal al presionar el botón y para la señal al soltarlo.



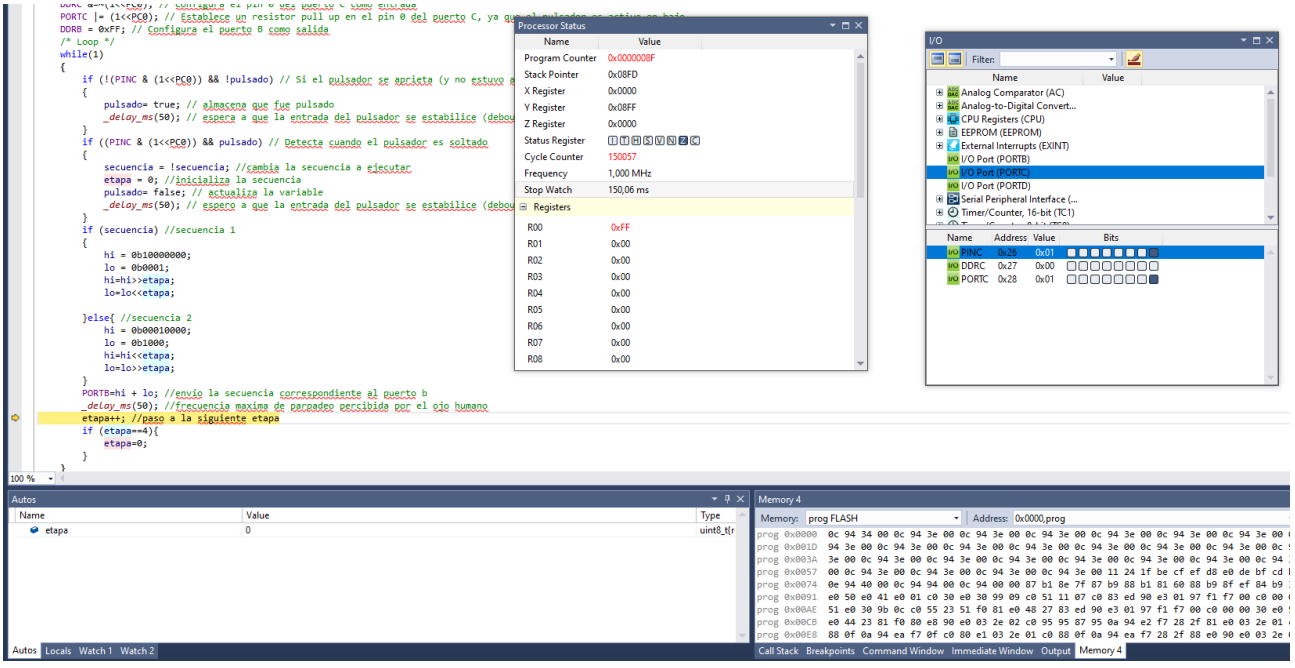
La inicialización toma 0.03 ms aproximadamente. PINC Tiene el valor 0x00 indicando que el pulsador está activado.



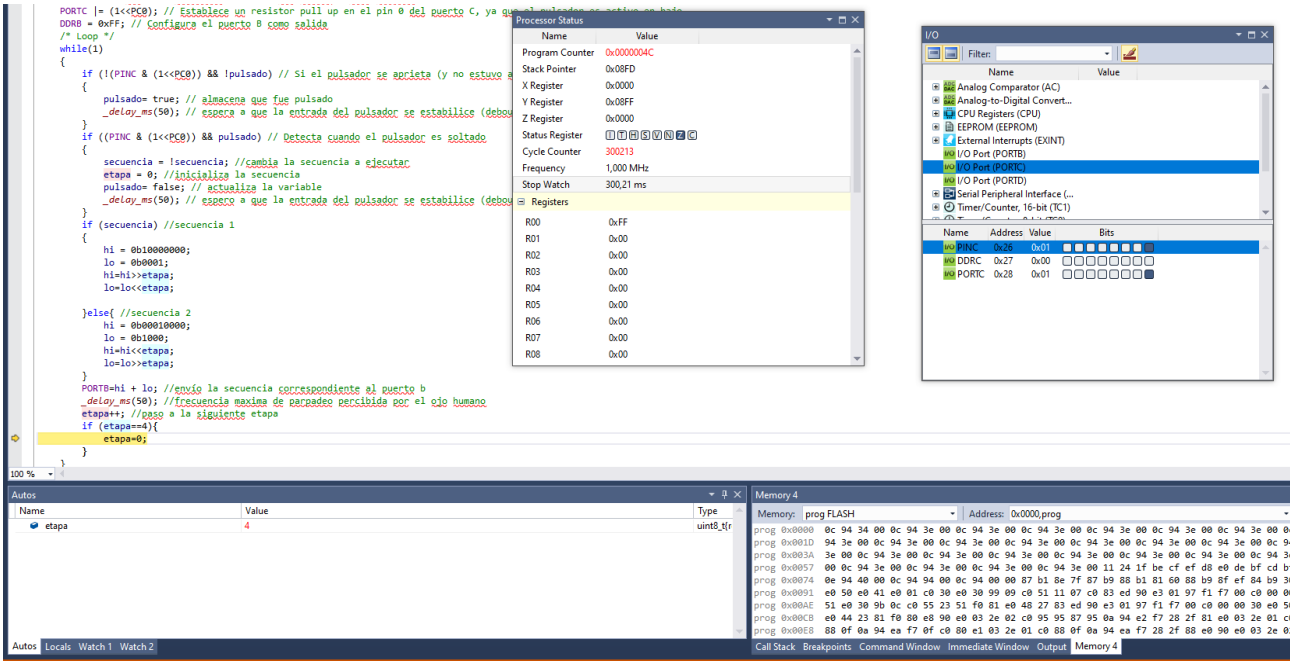
Durante el primer bloque if, la ejecución se detiene durante 50 ms. Se suelta el botón y PINC toma el valor 0x01.



Luego del segundo bloque if, debido al tiempo de debounce la ejecución se habrá detenido durante otros 50 ms adicionales.



La primera etapa finaliza con un tiempo de 150 ms.



La secuencia completa demora 300 ms, demostrando que, salvo la primera iteración, el resto dura 50 ms al igual que en el caso 1.

Se excluyeron del análisis los casos especiales que no son de interés para lo solicitado: existe la posibilidad de que no se detecte durante la primera iteración que el botón fue soltado, si este se suelta en la misma iteración pero después de ejecutarse la segunda sentencia if. Esto reduciría el tiempo de ejecución de la primera iteración y aumentaría el de la siguiente iteración en que el botón es soltado (debido a que se ejecutaría el segundo bloque if). El tiempo total resultaría el mismo.

4. Código

```
/*
 * Alternar Secuencia de LEDs
 * Created: 21/3/2022 14:40:10
 * Author : Adrian y Lautaro
 */

/* Inclusión de cabeceras de bibliotecas de código */
#include <avr/io.h> // Definición de Registros del microcontrolador
#define F_CPU 16000000UL // Especifico la frecuencia de reloj del MCU en 16MHz
#include <util/delay.h> // Retardos por software
#include <stdbool.h> // Variables booleanas

int main (void)
{
	/* Setup */
	bool secuencia = true; // La secuencia en ejecucion: true -> secuencia 1,
```

```
false -> secuencia 2
	bool pulsado = false; // Almacena si el pulsador fue apretado (se usa para reconocer cuando el pulsador es soltado)
	uint8_t etapa = 0; // Cada secuencia puede pensarse como un conjunto de 4 etapas, en la que alternan los pares de leds encendidos
	uint8_t hi; //Los 4 leds mas significativos
	uint8_t lo; //Los 4 leds menos significativos
	DDRC &=~(1<<PC0); // Configura el pin 0 del puerto C como entrada
	PORTC |= (1<<PC0); // Establece un resistor pull up en el pin 0 del puerto C, ya que el pulsador es activo en bajo
	DDRB = 0xFF; // Configura el puerto B como salida
	/* Loop */
	while(1)
	{
		if (!(PINC & (1<<PC0)) && !pulsado) // Si el pulsador se aprieta (y no estuvo apretado en la iteracion anterior)
		{
			pulsado= true; // almacena que fue pulsado
			_delay_ms(50); // espera a que la entrada del pulsador se estabilice (debounce)
		}
		if ((PINC & (1<<PC0)) && pulsado) // Detecta cuando el pulsador es soltado
		{
			secuencia = !secuencia; //cambia la secuencia a ejecutar
			etapa = 0; //inicializa la secuencia
			pulsado= false; // actualiza la variable
			_delay_ms(50); // espero a que la entrada del pulsador se estabilice (debounce)
		}
		if (secuencia) //secuencia 1
		{
			hi = 0b10000000;
			lo = 0b0001;
			hi=hi>>etapa;
			lo=lo<<etapa;

		}else{ //secuencia 2
			hi = 0b00010000;
			lo = 0b1000;
			hi=hi<<etapa;
			lo=lo>>etapa;
		}
		PORTB=hi + lo; //envío la secuencia correspondiente al puerto b
		_delay_ms(50);
		etapa++; //paso a la siguiente etapa
		if (etapa==4){
			etapa=0;
		}
	}
	return 0;
```

```
}
```