Universidad Nacional de La Plata

Trabajo Práctico 1

Control de periféricos externos con puertos de entrada/salida

Circuitos Digitales y Microcontroladores

Tarifa, Carla.
Carballo Ormaechea, Lucas.

Índice

1. Conexión de 8 Diodos al MCU

1.1.Enunciado

Se desea conectar 8 diodos LED de diferentes colores al puerto B del MCU y encenderlos con una corriente de 10mA en cada uno. Realice el esquema eléctrico de la conexión en Proteus. Calcule la resistencia serie para cada color teniendo en cuenta la caída de tensión VLED (rojo=1.8V, verde=2.2V amarillo=2.0V azul=3.0V) Verifique que la corriente por cada terminal del MCU no supere la capacidad de corriente de cada salida y de todas las salidas del mismo puerto en funcionamiento simultáneo.

1.2 Interpretación

En el programa de simulación Proteus , se creará un nuevo proyecto donde se utilizará un microcontrolador ATMEGA328P, y 8 diodos LEDs de distintos colores con sus respectivas resistencias.

Los diodos se conectan al PUERTO B del MCU, por lo que se tendrá que configurar como salida todo el puerto (desde PORTB0 a PORTB7). A su vez, se tendrá que calcular el valor de cada resistencia(Ley de Ohm), ya que cada color de diodo tiene un valor de voltaje diferente al resto.

Por último , deberán utilizarse amperímetros en cada diodo para verificar que no circule una corriente superior a 10mA.

1.3 Resolución

1.3.1 Agregado del MCU

El nombre del microcontrolador a utilizar es “ATMEGA328P”, el cual por defecto tiene una tensión de alimentación V_{cc} de 5V. En la Figura 1.1 se puede observar dicho microcontrolador.

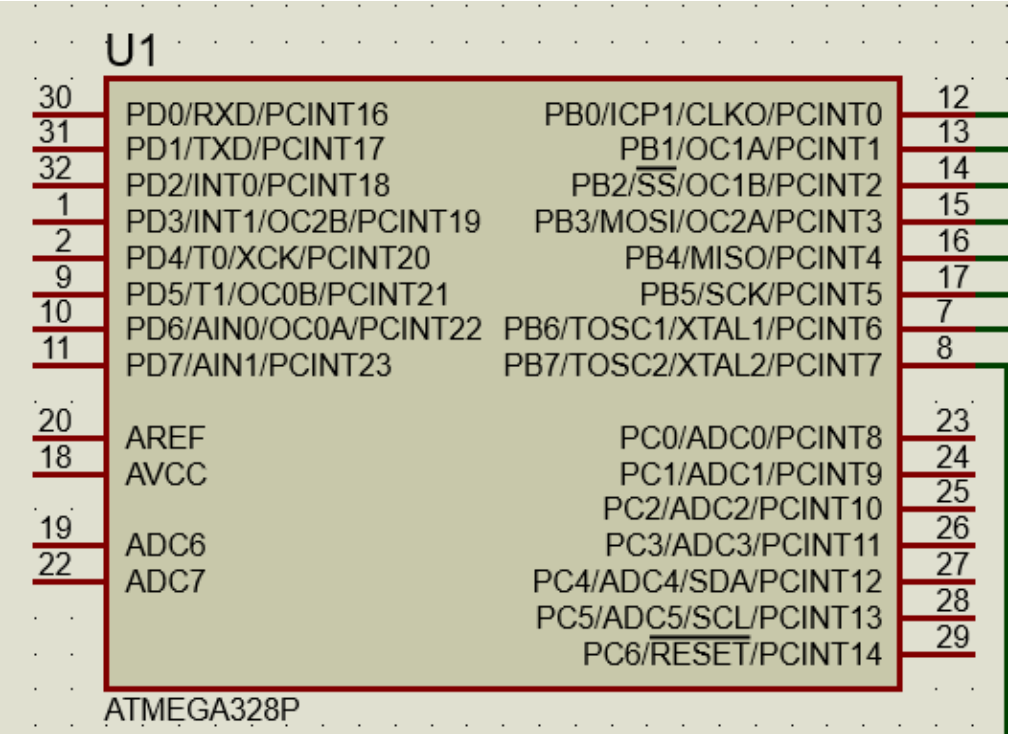

Figura 1.1. Microcontrolador Atmega328P en Proteus.

1.3.2 Agregado de Leds

Los colores de los diodos leds a utilizar son : amarillo,azul,blanco,naranja,morado,rojo,rosa y verde . En la Tabla 1.1 se encuentran los valores de caída de tensión de cada led y se puede observar que no superan los 5V de V_{cc}.

Description	Chemistry	# of Elements	Color Temperature (CCT Typ)	Peak Wavelength (λ / x-coord)	Dominant Wavelength (λ / y-coord)	Forward Voltage (Vf Typ)	Forward Voltage (Vf Max)
High Efficiency Red	GaP	2	~	700	660	2.0	2.5
Super Red	GaAlAs	3	~	660	640	1.7	2.2
Super Red	AlInGaP	4	~	660	640	2.1	2.5
Super High Intensity Red	AlnGaP	4	~	636	628	2.0	2.6
High Intensity Red	GaAsP	3	~	635	625	2.0	2.5
TS AlInGaP Red	AlInGaP	4	~	640	630	2.2	2.8
Super Orange	AlInGaP	4	~	610	602	2.0	2.5
Amber	GaAsP	3	~	605	610	2.0	2.5
Super Yellow	AlInGaP	4	~	590	588	2.0	2.5
TS AlInGaP Yellow	AlInGaP	4	~	590	589	2.3	2.8
Yellow	GaAsP	3	~	590	588	2.1	2.5
Super Ultra Green	AlInGaP	4	~	574	568	2.2	2.6
Green	GaP	2	~	565	568	2.2	2.6
Super Green	GaP	2	~	565	568	2.2	2.6
Pure Green	GaP	2	~	555	555	2.1	2.5
Ultra Pure Green	InGaN	3	~	525	520	3.5	4.0
Ultra Emerald Green	InGaN	3	~	500	505	3.5	4.0
Ultra Super Blue	InGaN	3	~	470	470	3.5	4.0
Ultra Violet	InGaN	3	~	410	~	3.5	4.0
Super Violet	InGaN	3	~	380	~	3.4	3.9
Turquoise	InGaN	3	~	0.19	0.41	3.2	4.0
Violet / Purple	InGaN	3	~	0.22	0.11	3.2	4.0
Pink	InGaN	3	~	0.33	0.21	3.2	4.0
Warm White	InGaN	3	3000K	~	~	3.3	4.0
Neutral White	InGaN	3	4000K	~	~	3.3	4.0
Cool White	InGaN	3	6000K	~	~	3.3	4.0

Tabla 1.1 Propiedades de los diodos Leds.

1.3.3.Agregado de resistores

A través de la Ley de Ohm , los valores proporcionados por la Tabla1.1 y los valores de provistos por la problemática , se calcula el valor de las resistencia de cada led con la siguiente fórmula:

$$R = \frac{V_{CC} - V_{LED}}{I}$$

-Vcc → es el voltaje de alimentación (puerto B del MCU).
-Vled → es la caída de tensión del LED.
-I → es la corriente deseada a través del LED (en este caso, 10 mA).

A continuación se muestra una tabla en donde se detalla el valor de tensión de cada led y el valor de su resistencia , obtenido con la fórmula mencionada anteriormente.

n° led	Color	$V_{Led}[V]$	R[**Ω**]
0	Azul	3.0	$\frac{5V-3.0V}{10mA}=200$
1	Verde	2.2	$\frac{5V-2.2V}{10mA}=280$
2	Naranja	2.0	$\frac{5V-2.0V}{10mA}=300$
3	Rojo	1.8	$\frac{5V-1.8V}{10mA}=320$
4	Amarillo	2.0	$\frac{5V-2.0V}{10mA}=300$
5	Rosa	3.2	$\frac{5V-3.2V}{10mA}=180$
6	Morado	3.2	$\frac{5V-3.2V}{10mA}=180$
7	Blanco	3.3	$\frac{5V-3.3V}{10mA}=170$

1.3.4 – Agregado de instrumentos

Se colocará un amperímetro en serie en cada diodo para poder verificar que la corriente por cada terminal del MCU no supere la capacidad de corriente de cada salida y de todas las salidas del mismo puerto en funcionamiento simultáneo.

1.4 – Simulación

La simulación consta de un pequeño programa, en donde los terminales del puerto B estén configurados como salida y que tengan un “1” en todos los bits del registro de datos . Estas condiciones se logran mediante dos sentencias :

```
DDRB = 0xFF;
PORTB=0xFF;
```

En la Figura 1.2 se puede observar el resultado de la simulación.

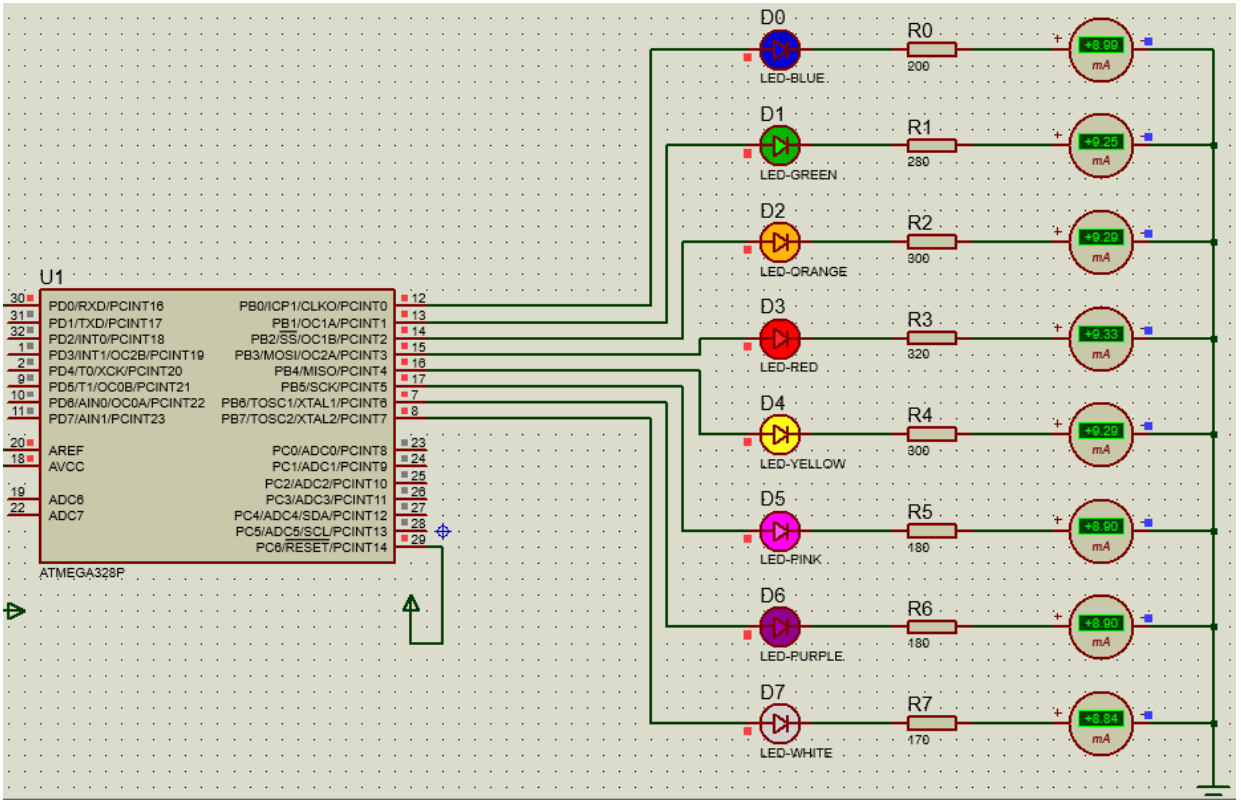

Figura 1.2. Simulación con los diodos encendidos en proteus

Como se puede observar en la Figura 1.2 , la corriente que circula por cada rama es menor a 10 mA .Esto se debe a que la tensión de salida del microcontrolador no es de 5V para todo valor de corriente, si no que entrega menor tensión a medida que aumenta la corriente.

2. Conexión del pulsador

2.1 Enunciado

Se desea conectar un pulsador a una entrada digital del MCU y detectar cuando el usuario presiona y suelta el pulsador. Muestra el esquema de conexión y determina la configuración del MCU que corresponda. Investigue sobre el efecto de rebote que producen los pulsadores e implemente un método para eliminar este efecto en su algoritmo de detección (puede encontrar información útil en la bibliografía).

2.2 Interpretación

Se utilizará un pulsador, el cual será el encargado de dar comienzo a la secuencia de encendido de los bits. Cabe destacar que habrá efecto rebote al

pulsarlo. El fenómeno mencionado provoca que el contacto mecánico del pulsador rebote entre su estado abierto y cerrado varias veces en un corto periodo de tiempo, antes de estabilizarse en su estado final. Para solucionar esto, se aplicará un circuito anti-rebote RC.

2.3 Resolución

En proteus, se conecta el pulsador a un circuito RC, el cual va a estar conectado al puerto D, el mismo se va a configurar en el software como entrada. Cabe destacar que solo se utilizara el pin 0 de este puerto.
Para el efecto rebote se aplicará al circuito una resistencia y un capacitor, ya que al conectar un capacitor al pulsador actúa como un filtro de rebotes, este se carga y descarga, suavizando las transiciones de estado y evitando que se registren como pulsaciones reales. Más adelante, se detalla la configuración de esta solución en la parte del software.

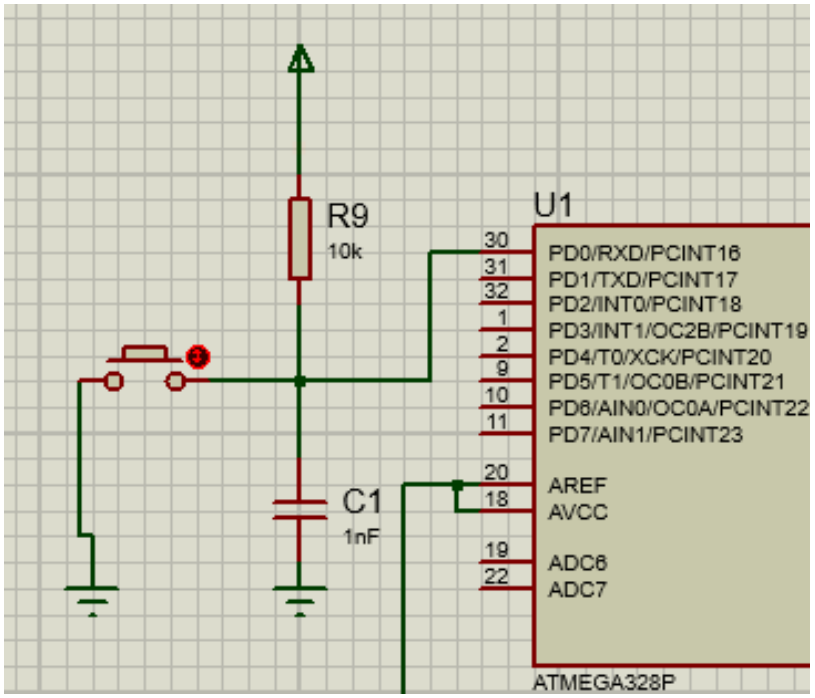

Figura 2.1 Conexión RC con pulsador para evitar el efecto rebote.

En la Figura 2.2 se muestra el diseño completo de las dos partes especificadas, el pulsador y los 8 diodos con sus respectivas resistencias.

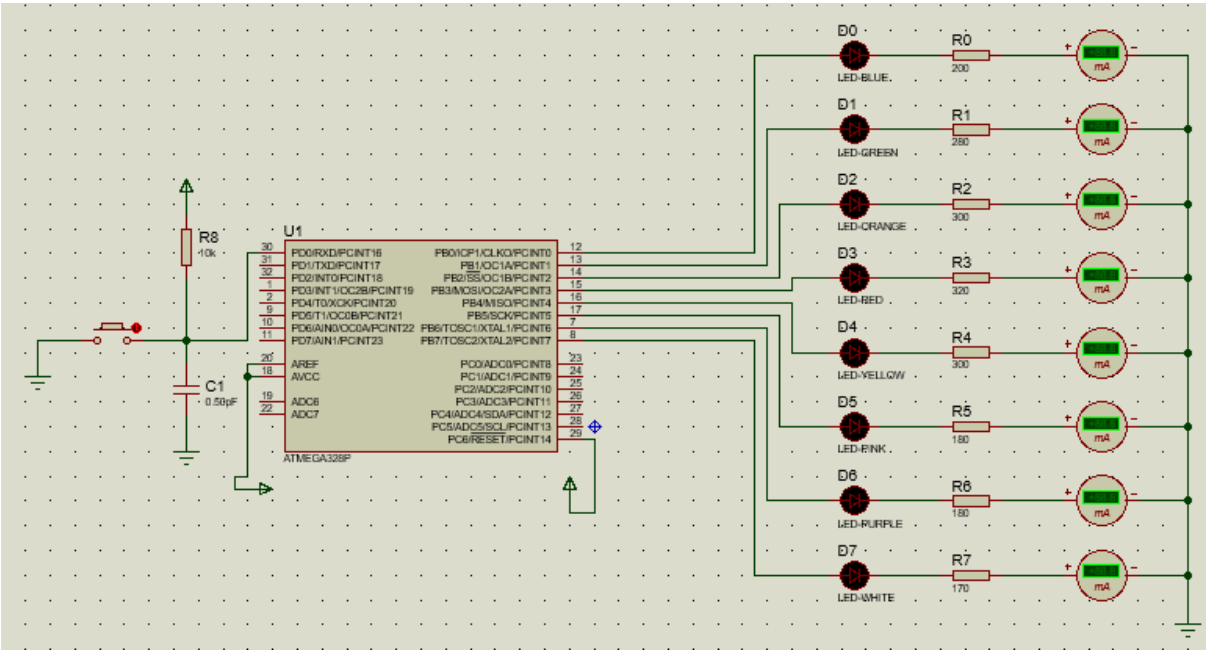

Figura 2.2. Diseño completo

3. Diseño de software

3.1 Enunciado

Realice el programa para que el MCU encienda los LEDs del puerto B con la siguiente secuencia de encendido repetitiva: b0 y b7 – b1 y b6 – b2 y b5 – b3 y b4. Luego, cuando el usuario presione y suelte el pulsador debe cambiar a la secuencia: b7-b6-b5-b4-b3-b2-b1-b0. Si presiona y suelta nuevamente vuelve a la secuencia original y así sucesivamente. Elija un retardo adecuado para la visualización en el simulador. Justifique.

3.2 Interpretación

Se deberá realizar un programa en C que permita ir encendiendo los diodos conectados en el ejercicio 1.Por defecto, al iniciar la ejecución la secuencia es b0 y b7 – b1 y b6 – b2 y b5 – b3 y b4, la cual se repetirá hasta que el usuario presione y suelte el botón . Una vez que el usuario haya soltado el botón , cambia la secuencia a b7-b6-b5-b4-b3-b2-b1-b0 y si el usuario vuelve a presionar y soltar el botón , deberá cambiar la secuencia nuevamente.

3.3 Solución

3.3.1 Implementación de secuencias

La primera secuencia de encendido de Leds se puede entender como 2 bits “1” que se desplazan hacia la izquierda . Uno de los bits “1” rotará entre los bits b0,b1,b2 y b3 , y el otro bit “1” rotará entre los bits b7,b6,b5 y b4.

A través de una variable “posición (i/j)” , se indicará el encendido en simultáneo de los leds y la segunda secuencia , utilizara la misma variable “posición” pero se encenderá un led a la vez empezando por el bit b7 hacia el bit b0.

Para el cambio de secuencias, se utilizará una variable (“ESTADO”), que dependiendo el valor que tenga, se realiza una secuencia específica. Esta variable se modifica cada vez que se presiona el pulsador.

3.3.2 Pseudocódigo

```
Importación de librerías.
Inicio
    Inicializar como salida el puerto B
    Inicializar como entrada el puerto D0
    Leds apagados del puerto B.
    Ponemos el estado en 0.
Inicio ciclo repetitivo.
    Si estamos en el estado 0
        Tiempo Espera
        Mientras el estado sea igual a 0
            Enciendo los pines con la secuencia 0,7 - 1,6 - 2,5 -
        3,4
            Tiempo Espera
            Si el pulsador fue presionado
                Cambió el Estado a 1.
                Tiempo Espera

    Sino
        Mientras Estado sea igual a 1
            Enciendo los pines con la secuencia 7,6,5,4,3,2,1,0
            Tiempo Espera
```

Si el pulsador fue presionado
Cambió el Estado a 0.
Tiempo Espera

Fin

Se aplicará un tiempo de espera de 250 ms, para que el programa funcione correctamente, ya que con menos tiempo de retardo no se logra apreciar bien el funcionamiento del mismo, y más tiempo hace obsoleto el programa (mucho tiempo esperando).

3.3.3 Simulación

En las siguientes figuras se pueden observar las diferentes secuencias simuladas en proteus

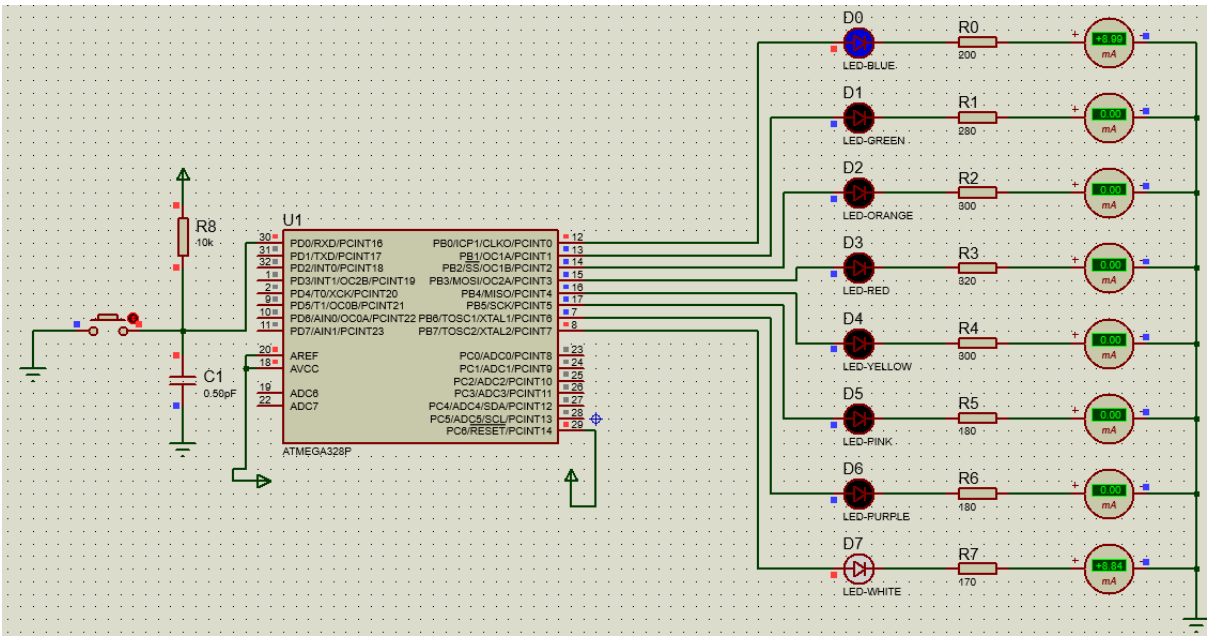

Figura 3.1. Secuencia 1 , antes de presionar el pulsador

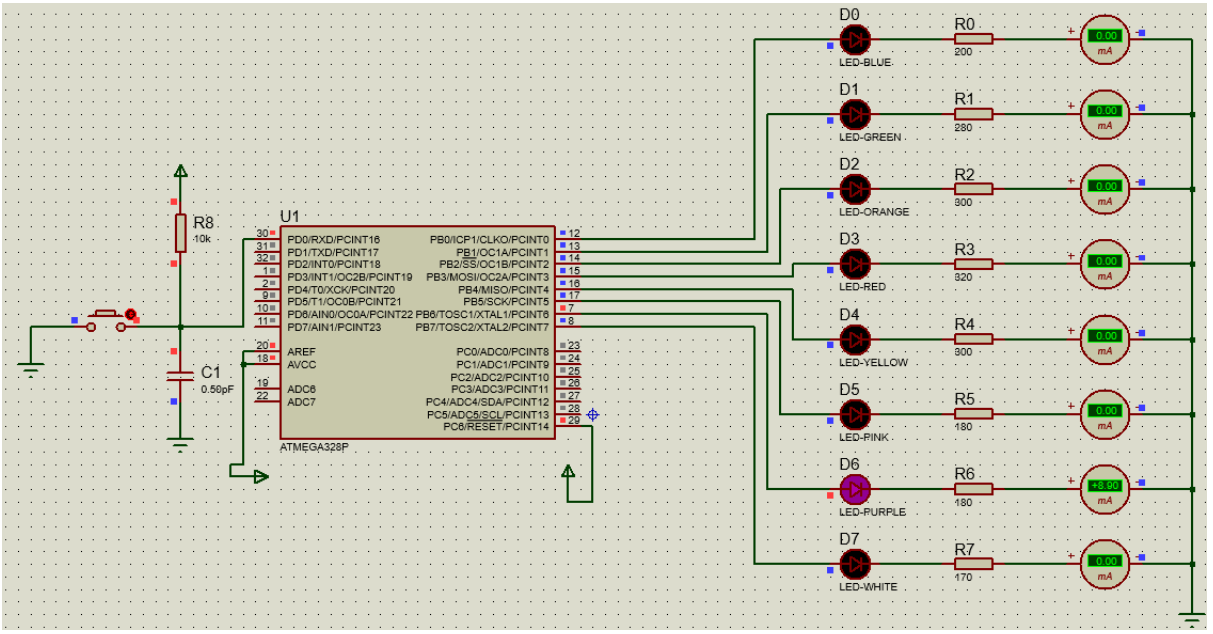

Figura 3.2. Secuencia 2 , luego de presionar el pulsador.

A continuación , se encuentra el funcionamiento del programa en microchip. Como no contamos con una herramienta física, procedemos a simularlo. Tocamos Step intro para ver la reacción de nuestros registros.

```
int main(void)
{
    // 1 -> Salida|| 0 -> Entrada
    DDRB =  0xFF; //Todo PORTB es salida
    DDRD &= ~(1<<PORTD0);  //Bit 0 de PORTD es entrada
    PORTB = 0b00000000; //Inicio con leds apagados
    int ESTADO = 0;
    while (1) //Buce infinito
    {
        if (ESTADO == 0) // Inicia con estado en 0
        {
            _delay_ms(250); //retardo 250 ms
            while (ESTADO == 0) // Inicia secuencia 1
            {
                for (int j=0; j<=3; j++)
                {
                    PORTB = (1<<j) | (1<<(7-j));
```

Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	

Name	Address	Value	Bits
PINB	0x23	0x00	
DDRB	0x24	0xFF	
PORTB	0x25	0x00	

```
    // 1 -> Salida|| 0 -> Entrada
    DDRB =  0b11111111; // Config como salida des
    DDRD =  0b11111110; //Config como entrada el
    PORTB = 0b00000000; // Inicio con todo apagad
    int ESTADO = 0;
    while (1) //Bucle infinito
```

Serial Peripheral Interface (...

Name	Address	Value	Bits
PIND	0x29	0x00	
DDRD	0x2A	0xFE	
POR...	0x2B	0x00	

3.3.4 Código

```
#include <avr/io.h>
#define F_CPU 20000000UL // Frecuencia de 20MHz
#include <util/delay.h> //Libreria de retardos
#define BUTTON_STATE() (PIND&(1<<PIND0))

int main(void)
{
    // 1 -> Salida|| 0 -> Entrada
    DDRB =  0xFF; //Todo PORTB es salida
    DDRD &= ~(1<<PORTD0);  //Bit 0 de PORTD es entrada
    PORTB = 0b00000000; //Inicio con leds apagados
    int ESTADO = 0;
    while (1) //Buce infinito
    {
        if (ESTADO == 0) // Inicia con estado en 0
        {
            _delay_ms(250); //retardo 250 ms
            while (ESTADO == 0) // Inicia secuencia 1
            {
                for (int j=0; j<=3; j++)
                {
                    PORTB = (1<<j) | (1<<(7-j));
                    _delay_ms(250);
                    if (BUTTON_STATE()!=1) // Verifica si el pulsador fue presionado
                    {
                        _delay_ms(250); // 250 ms
                        ESTADO = 1;
                    }
                }
            }
            } else {
                  while (ESTADO == 1)//Inicia secuencia 2
                  {
                    for (int i=7; i>=0; i--){
                        PORTB = (1<<i);
                        _delay_ms(250);
                        if (BUTTON_STATE()!=1)
                          {
                            _delay_ms(250);
                             ESTADO = 0;
                          }
                     }

                  }

              }

        }
    }
```

4. Conclusión

4.1.Enunciado

Saque conclusiones sobre el funcionamiento del programa, sobre las ventajas y desventajas de utilizar retardos (delays) para temporizar acciones y cómo estos afectan el tiempo de respuesta a la acción del usuario. Hágase preguntas como por ejemplo: ¿qué sucede si se deja presionado constantemente el pulsador? ¿Qué sucede si el retardo es de 1 segundo?

4.2 Desarrollo

El funcionamiento del programa cumple con lo pedido en los enunciados. Al haberse utilizado un algoritmo con retardo se obtuvo una solución rápida debido a que la función delay ya se encuentra creada y se adecua a la frecuencia de reloj especificada pero el MCU se encuentra “demorado” hasta que se cumpla el tiempo de delay , esto provoca que si se presiona y suelta rapido el pulsador pase desapercibido o si se mantiene presionado el pulsador por mucho tiempo cambie la secuencia por más que no se haya soltado el pulsador.