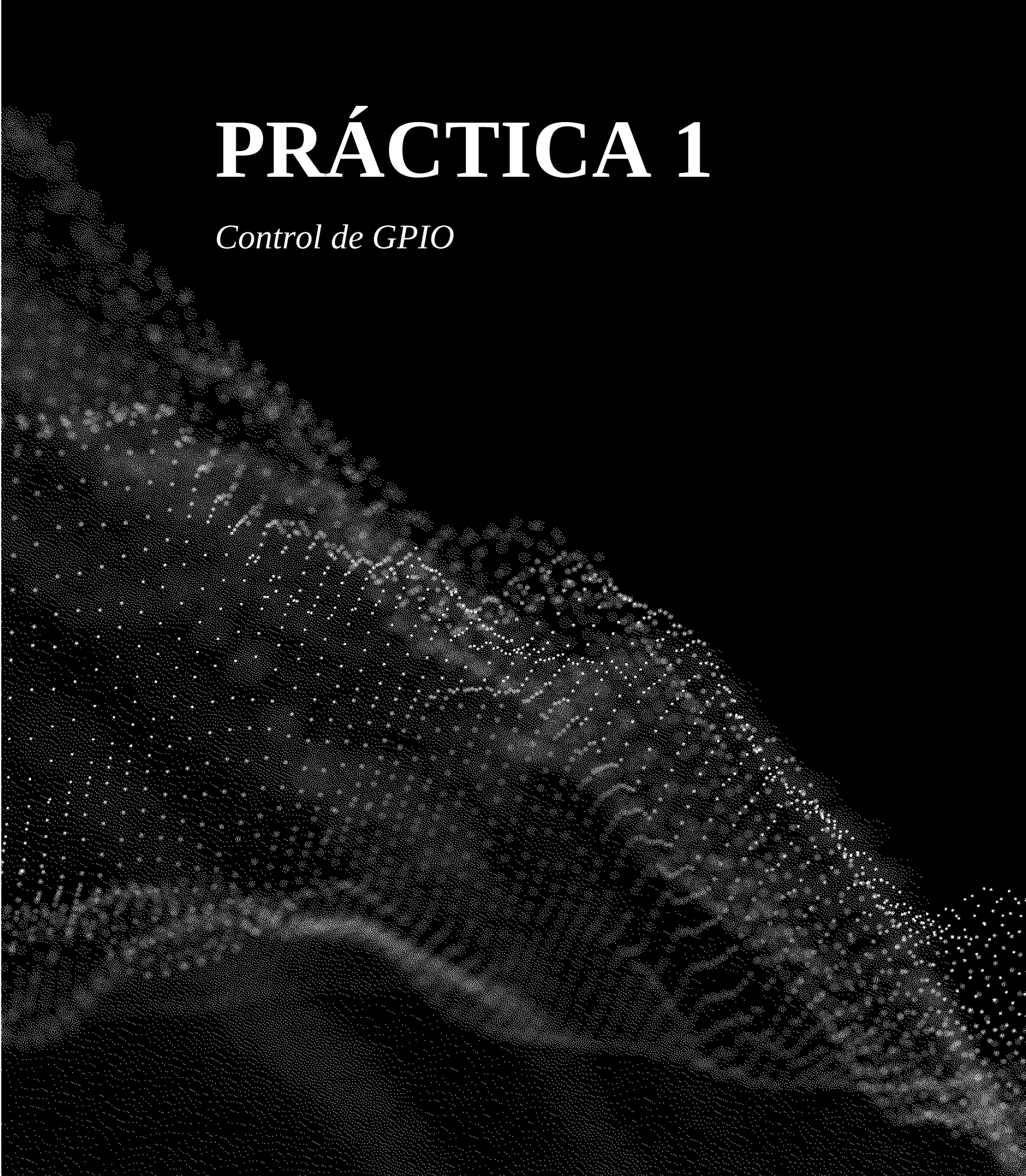
PRÁCTICA 1

Control de GPIO



María González Herrero

Santiago Molpeceres Díaz

INDICE

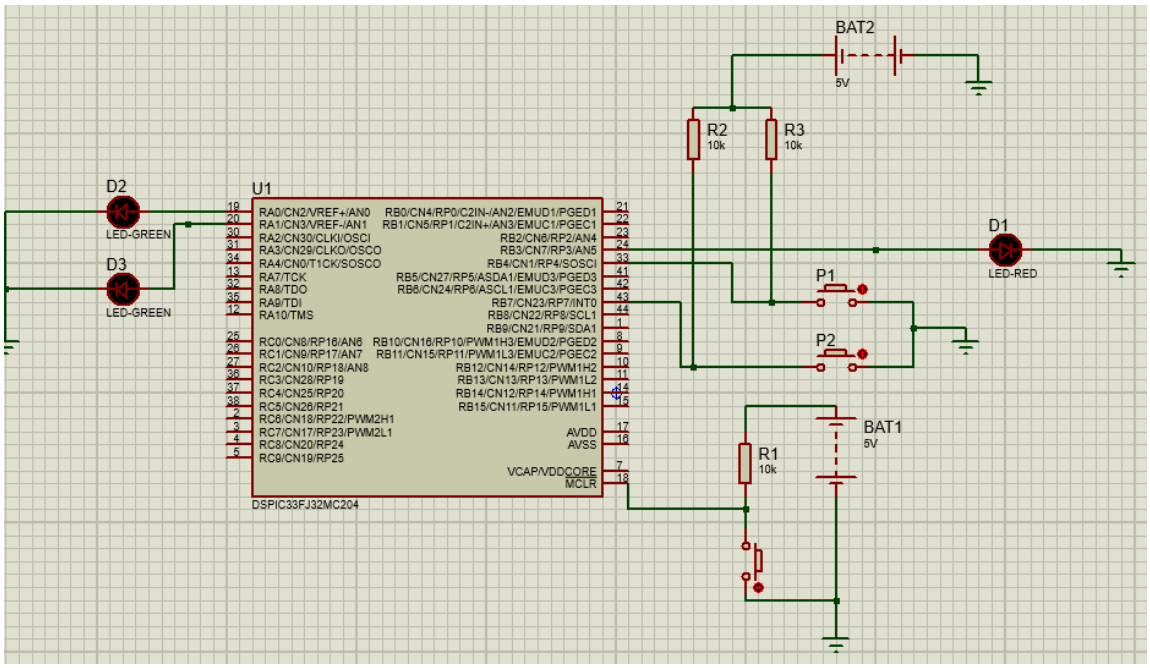
PRÁCTICA 1

En esta primera práctica de la asignatura de Sistemas empotrados hemos aprendido a realizar la comunicación entre nuestro microprocesador dsPIC33FJ32MC204 con botones y leds para ver su comportamiento, como el parpadeo o mantener el led encendido apretando un botón.

Enunciado

1. Lograr que el diodo LED D1 parpadee a un ritmo de 1Hz al ejecutar el programa. Considerad una frecuencia de la CPU de 4MHz.
2. Por defecto los diodos LED D2 y D3 deben permanecer apagados. En el momento de pulsar el botón P1, los diodos D2 y D3 debe encenderse durante el tiempo que P1 permanezca pulsado. En caso contrario deben permanecer apagados.
3. Si se mantiene presionado el pulsador P2, el diodo D3 debe parpadear a un ritmo de 2 Hz. En caso contrario, el diodo D3 debe permanecer apagado. A su vez, se debe cumplir en todo momento el requisito del punto 1. (Considerad una frecuencia de CPU de 4MHz).

Circuito



Parpadeo LED D1-Luz Roja

Este es primer ejercicio de la práctica:

```
#define LED_ROJO LATBbits.LATB3

void enciende_Apaga_Rojo(){

    LED_ROJO = !PORTBbits.RB3;

}

  //EJERCICIO 1

   enciende_Apaga_Rojo();

   delay_ms(500);
```

Lo que tenemos que hacer es que parpadee a un ritmo de 1Hz nuestro LED-ROJO.

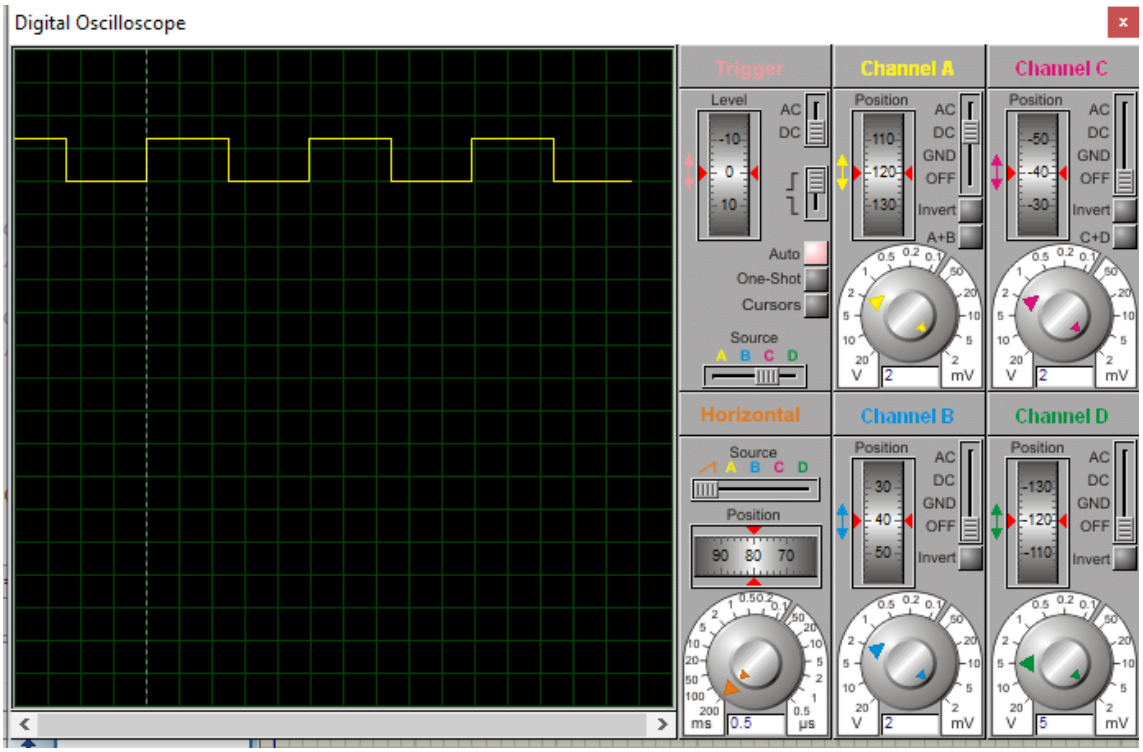
Primero, para ser más legible nuestro código hemos definido nuestra salida delRB3 como LED_ROJO.

Después hemos creado una función llamada *"enciende_Apaga_Rojo"*, que es la encargada de hacer que el LED_ROJO parpadee, haciendo que se invierta la entrada de RB3 (donde está situado nuestro led).

Finalmente, en el ejercicio hemos llamado a nuestra función y le hemos dado un tiempo de 500ms, que con la función de "delay_ms", creará un retardo de 1Mhz.

Simulación (oscilador)

Con el programa de Proteus a través del oscilador hemos querido mostrar una simulación del comportamiento de nuestro ejercicio, como se puede ver a continuación:



La línea amarilla que se observa es led rojo parpadeando, cada vez que parpadea se crea un ciclo de reloj de 1Hz.

Pulsar botón P1 para los LEDS D2 y D3

A continuación, podemos ver el segundo ejercicio de la práctica:

```
#define LED_VERDE LATAbits.LATA0

#define LED_VERDE2 LATAbits.LATA1

//EJERCICIO 2
if(PORTBbits.RB4 == 0)
{
   LED_VERDE = 1;//ON LED_VERDE (D2)
   LED_VERDE2 = 1;//ON LED_VERDE2 (D3)
} else {
   LED_VERDE = 0;//OFF LED_VERDE (D2)
   LED_VERDE2 = 0;//OFF LED_VERDE2 (D3)
}
```

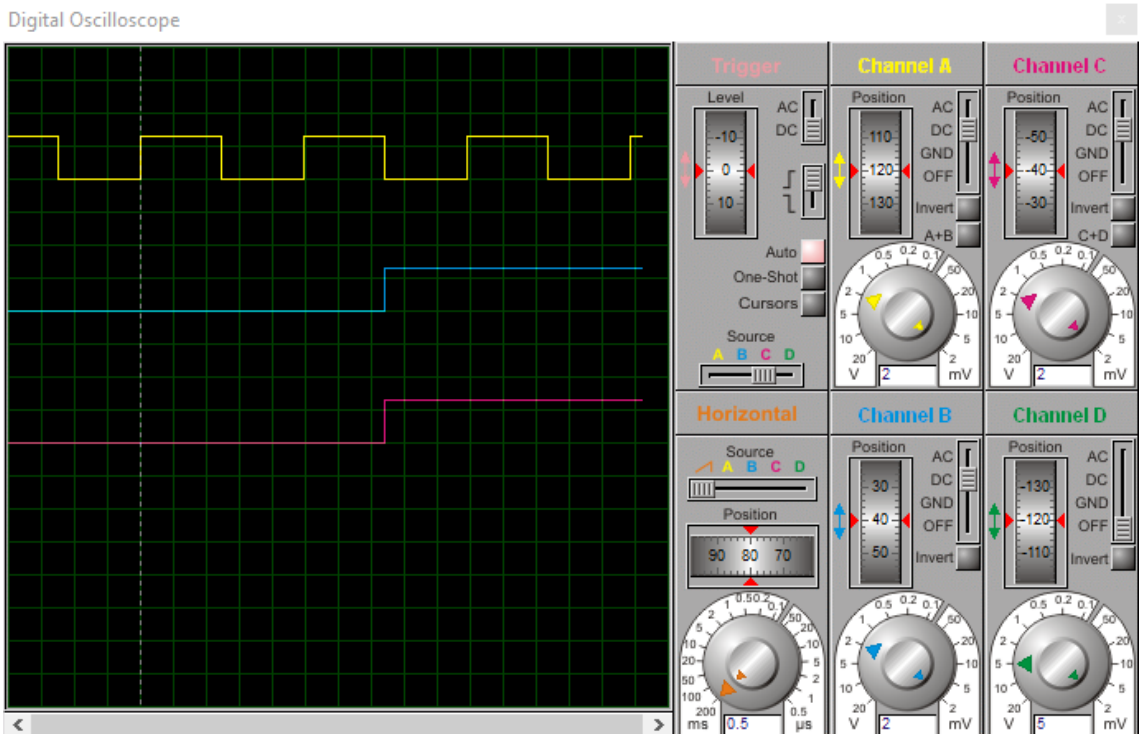
Tenemos que apretar el botón P1 (RB4) y se encienden los LEDS D2 y D3, al dejar de pulsar el botón P1 se apagarán los dos LEDS.

Al igual que en el anterior ejercicio, hemos querido hacer nuestro código más legible poniendo nombres a nuestros 2 leds.

Después hemos hecho una condición en la cual si a nuestro botón P1 (RB4) le pulsamos ambos leds cambiarán su valor a “1”, es decir, se encenderán y en caso contrario a “0”, lo cual significa que se apagarán.

Simulación (oscilador)

Con el programa de Proteus a través del oscilador hemos querido mostrar una simulación del comportamiento de nuestro ejercicio, como se puede ver a continuación:



La línea azul es la correspondiente al LED_VERDE (D2) y la línea rosa es correspondiente al LED_VERDE(D3). Como se puede ver en la simulación al principio el botón P1 no está pulsado por lo que los leds están apagados, en el momento de pulsar el botón P1, ambos leds (D2 y D3) se activan, cambiando su valor a 1.

Parpadeo LED D3, con el botón P2

Por último, el tercer ejercicio en el cual pulsando el botón P2 nuestro led-verde (D3) parpadea a una frecuencia de 2Hz y si se deja de pulsar se apaga.

```
void enciende_Apaga_Verde_Rojo(){

    LED_ROJO = !PORTBbits.RB3;

    LED_VERDE2 = !PORTAbits.RA1;

}
```

Primero hemos reutilizado la parte de cambiar el nombre a las variables para hacer código más legible.

Además hemos creado una función llamada "*enciende_Apaga_Verde_Rojo();*", la cual se encarga de hacer parpadear tanto al led-rojo (D1) como al led-verde (D3).

A parte hemos reutilizado la función de "*enciende_Apaga_Rojo();*", que ya creamos para el ejercicio 1. Para así completar el ejercicio 3 de la forma que se muestra a continuación:

```
//EJERCICIO 3
if(PORTBbits.RB7==0){
    enciende_Apaga_Verde_Rojo();
    delay_ms(500);
    enciende_Apaga_Rojo();
    delay_ms(500);
    enciende_Apaga_Verde_Rojo();
    delay_ms(500);
    enciende_Apaga_Rojo();
    delay_ms(500);
}else{
    enciende_Apaga_Rojo();
    delay_ms(500);
}
```

Cuando pulsamos P2 (RB7) se hace parpadear el verde con el doble de ciclos de reloj (2Hz) que el rojo, el cual se mantiene al valor del ejercicio 1 (1Hz), en caso contrario, en el cual dejemos de pulsar el botón de P2, seguirá parpadeando el led-rojo como lo haría normalmente.

Explicación del código

Si pulsamos el botón P2->

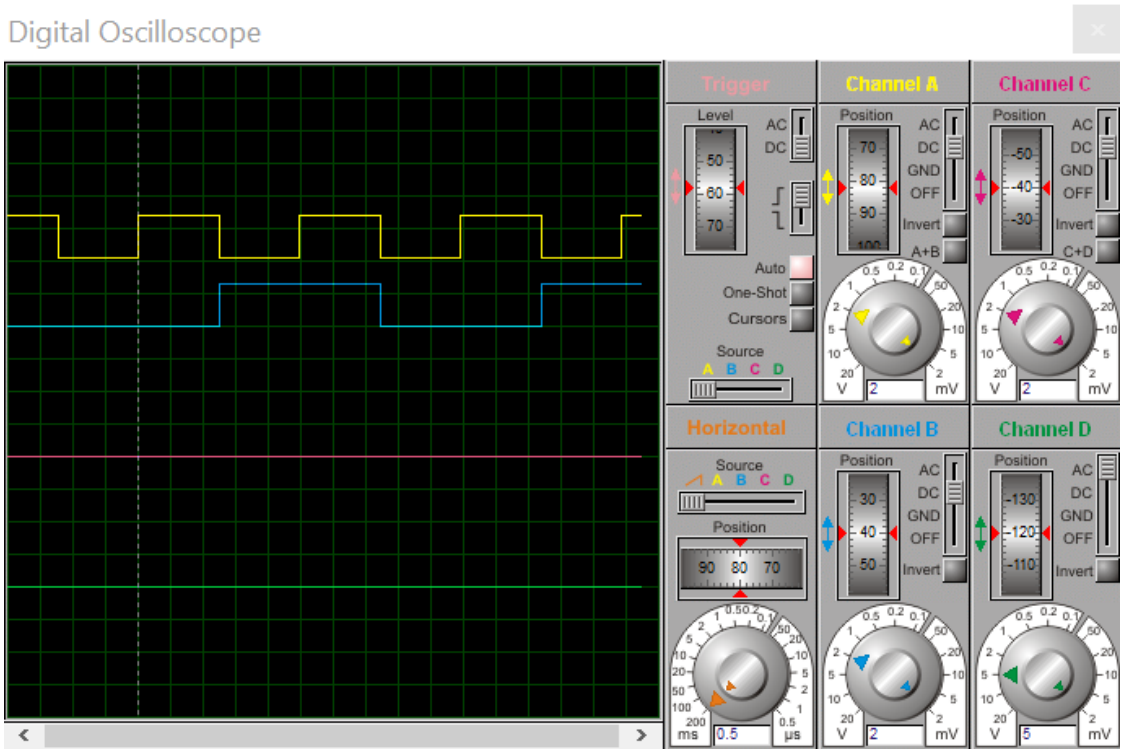
- Cambiamos los valores de los leds D1 y D3 (Rojo y Verde2).
- Generamos un delay de 1MHz.
- Cambiamos el valor únicamente del Led Rojo, ya que su pulso debe ser de 1MHz.
- Generamos un delay de 1MHz -> Como suma del anterior, ya tendríamos 2Mhz.
- Cambiamos los valores de los leds D1 y D3 (Rojo y Verde2), ya que D1, se actualiza cada 1MHz, y D3 se actualiza cada 2MHz, que es la suma de los anteriores.
- Generamos un delay de 1MHz
- Cambiamos el valor únicamente del Led Rojo, ya que su pulso debe ser de 1MHz.
- Generamos un Delay de 1MHz -> como suma del delay anterior, tendríamos 2MHz. Por eso como consecuencia, si sigue el botón pulsado, volverá a cambiar el valor de los leds D1 y D3, produciéndose así un buble cumpliendo los márgenes de parpadeo de cada led.

Si no pulsamos el botón P2->

- El led D1 (Rojo) Parpadeará **siempre** con la frecuencia de 1MHz.

Simulación (oscilador)

Con el programa de Proteus a través del oscilador hemos querido mostrar una simulación del comportamiento de nuestro ejercicio, como se puede ver a continuación:



La línea azul es la correspondiente al LED_VERDE (D2), el cual se puede observar cómo al pulsar el botón P2 se inicializa y con cada dos pulsos de reloj del LED_ROJO (D1), es un único pulso de reloj del LED_VERDE (D2).

Función Delay

La función delay, como su nombre indica, generará un retardo en orden de megahercios (Mhz).

Para este programa hemos usado un bucle while.

```
void delay_ms (unsigned long time_ms)
{
    unsigned long u=0;
    while(u<time_ms*450){
        asm("NOP");
        u++;
    }
}
```

Esta función recibe un parámetro, el cual es el tiempo que queremos pausarnos en milisegundos(ms).
Este número multiplicado por la constante (450), será el máximo de tiempo de retardo.

450-> es una constante llamada “Parámetro de Calibración” para que el retardo sea aproximado a lo que sería el requisito real.