UNIVERSIDAD DE SANTIAGO DE CH
FACULTAD DE INGENIERÍA
Departamento de Ingeniería Mecánica
Área de Procesos Mecánicos

Leds Asíncronos

Se desea mantener tres leds encendidos de forma intermitente de manera simultanea e independiente, cuyos periodos de funcionamiento puedan ser escogidos de una lista {0.05,0.1,0.15,0.4}. Se debe ser capaz de ajustar la frecuencia de cada led de forma individual (Switch 2), para luego cambiar la frecuencia del led seleccionado desde la lista de frecuencias mostrada anteriormente (Switch 1). Los botones se ubican en el puerto C y los leds en el puerto D del microcontrolador

Solución:

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define N_Procesos 5
#define Frec_inicial 781*0.5
unsigned char S1_deb = 0;
unsigned char S2_deb = 0;

unsigned char freq_pointer[3] = {0,0,0};
float frecuencias[4] = {0.05,0.1,0.15,0.4};
unsigned char led_select = 0;


typedef struct//Define estructura de un proceso
{
	unsigned char IDTASK;
	unsigned long TIMER;
	unsigned long DELTA;
} PROC_STRUCT;
static volatile PROC_STRUCT Procesos[N_Procesos];//Crea numero de procesos

void config_procesos()
{
	Procesos[0].DELTA = 20;//aprox 13ms
	Procesos[1].DELTA = 20;
	Procesos[2].DELTA = Frec_inicial;
	Procesos[3].DELTA = Frec_inicial;
	Procesos[4].DELTA = Frec_inicial;


	for(unsigned char i=0;i<=N_Procesos;i++)
	{
		Procesos[i].IDTASK = 0;
		Procesos[i].TIMER = 0;
	}
	Procesos[2].IDTASK = 0;
	Procesos[3].IDTASK = 0;
	Procesos[4].IDTASK = 0;
```

```
}

void setup_timer()
{
	TCCR0B |=_BV(CS01);//Prescaler 8
	TIMSK0 |= _BV(TOIE0);//Timer overflow interrupt enable
	sei();
}
void setup_hardware()
{
	DDRD = 255; //Todo como salida
	PORTC = 255; //Todo con pull up
}

ISR(TIMER0_OVF_vect)//Interrupcion overflow Timer0
{
	for(unsigned char i=0;i <= N_Procesos-1;i++)
	{
		Procesos[i].TIMER++;//Suma 1 a .Timer en cada proceso
	}
}

int main(void)
{
    setup_timer();
	setup_hardware();
	config_procesos();
    while (1)
    {

		if(Procesos[0].TIMER >= Procesos[0].DELTA)//Selección de frecuencia
		{
			Procesos[0].TIMER = 0;
			if(bit_is_clear(PINC, PC0))
			{
				if(S1_deb < 250)
				{
					S1_deb++;
				}
			} else {
				if(S1_deb > 35)
				{
					//Hacer algo
					freq_pointer[led_select]++;
					if(freq_pointer[led_select] >=4)
					{
						freq_pointer[led_select] = 0;
					}
					Procesos[led_select+2].DELTA
=(frecuencias[freq_pointer[led_select]]*7812);
				}
				S1_deb = 0;
			}
		}


		if(Procesos[1].TIMER >= Procesos[1].DELTA)//Selección de frecuencia
		{
			Procesos[1].TIMER = 0;
			if(bit_is_clear(PINC, PC1))
			{
				if(S2_deb < 250)
```

```
                        {
                                S2_deb++;
                        }
                        } else {
                        if(S2_deb > 35)
                        {
                                //Hacer algo
                                led_select++;
                                if(led_select >2)
                                {
                                        led_select=0;
                                }

                        }
                        S2_deb = 0;
                }
        }


        if(Procesos[2].TIMER >= Procesos[2].DELTA)
        {
                Procesos[2].TIMER = 0;
                PORTD ^= _BV(PD0);
//              if(Procesos[2].IDTASK == 0)
//              {
//                      PORTD |= _BV(PD0);
//                      Procesos[2].IDTASK = 1;
//              } else if (Procesos[2].IDTASK == 1)
//              {
//                      PORTD &=~_BV(PD0);
//                      Procesos[2].IDTASK = 0;
//              }
        }

        if(Procesos[3].TIMER >= Procesos[3].DELTA)
        {
                Procesos[3].TIMER = 0;
                PORTD ^= _BV(PD1);
        }

        if(Procesos[4].TIMER >= Procesos[4].DELTA)
        {
                Procesos[4].TIMER = 0;
                PORTD ^= _BV(PD2);
        }
    }
}
```