

# CT1

Cuadernos técnicos  
de FPGA's Libres



Autor: (Obijuan)

**Juan González-Gómez**

# Unidad de PWM de frecuencia aproximada

En este cuaderno aprenderemos cómo hacer  
nuestras propias unidades de PWM y cómo  
usarlas en nuestras aplicaciones.



[fpgawars.github.io](https://fpgawars.github.io)

# Unidad de PWM de frecuencia aproximada

Autor

Juan González-Gómez  
(Obijuan)


Contacto:

@Obijuan\_cube  
<https://github.com/obijuan>

FPGA WARS:

[www.fpgawars.github.io](http://www.fpgawars.github.io)  
[info@fpgawars.github.io](mailto:info@fpgawars.github.io)

Licencia:



Colección:

Jedi-v1.1.0.zip  
Para descargar e instalar [pinche aquí](#)

Ejemplos:

Todos los ejemplos de este cuaderno técnico están en este repositorio: [Ejemplos CT1: Unidad de PWM de frecuencia aproximada](#).  
  
También encontrarás una selección de ellos en el menú Archivo/Ejemplos/PWM de Icestudio, una vez instalada y seleccionada la colección Jedi

Enlaces:

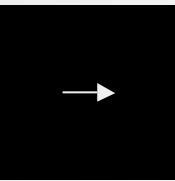
Tutoriales:  
[Electrónica digital para makers con FPGAs Libres](#)

VideoBlog 26:  
[Probando el analizador lógico compatible Salee, con la FPGA](#)

Tutoriales Verilog:  
[Diseño Digital para FPGAs, con herramientas libres](#)

CONTENIDO

Cuaderno Técnico 01



INTRODUCCIÓN	4
FRECUENCIA	6
CONTADORES: DIVISORES DE FRECUENCIA	11
ESTRUCTURA DE LA UNIDAD PWM	16
APLICACIÓN: CONTROL DEL BRILLO DE UN LED	18
BLOQUE DE PWM	26
EJEMPLOS DE USO DEL BLOQUE PWM	31



# Introducción



OBIJUAN

Ingeniero de teleco y  
doctor en robótica

En los sistemas digitales sólo tenemos dos niveles de tensión: 0 y 1. Si lo aplicamos a un LED, en principio sólo podríamos hacer que el LED esté apagado (0) o luciendo a su máxima intensidad (1)

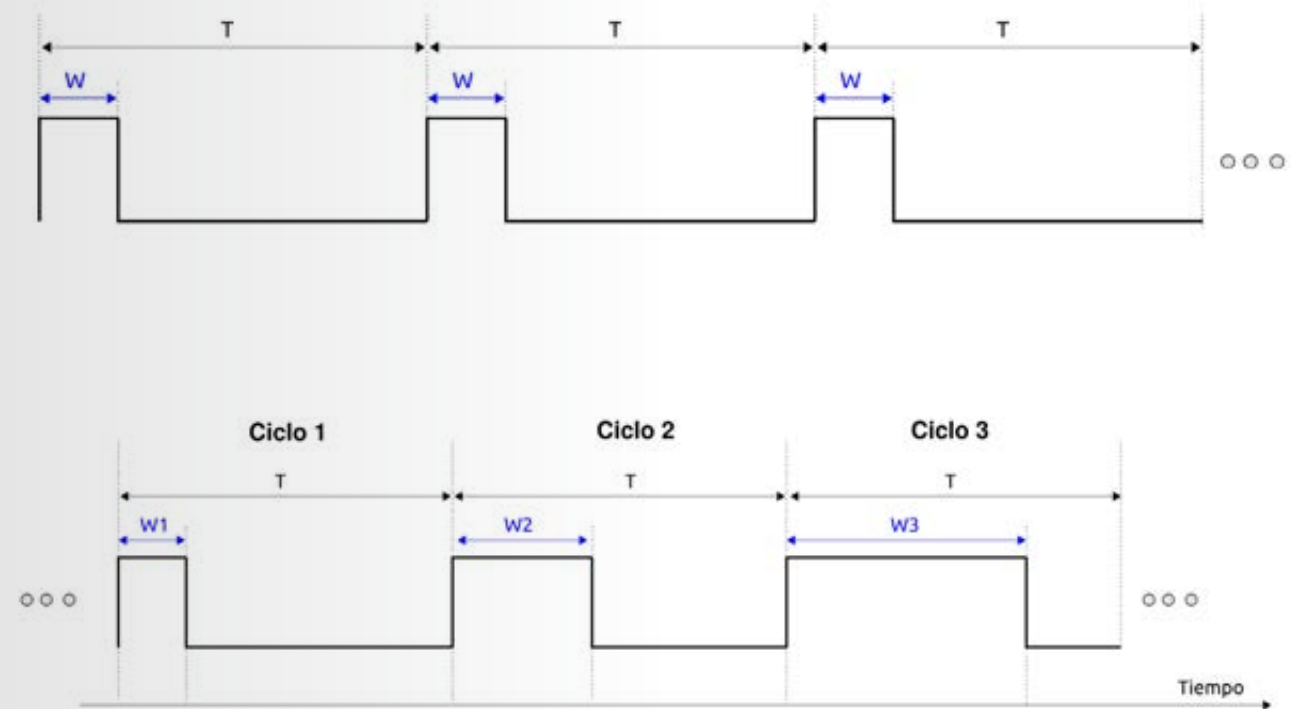
La Modulación por anchura de pulsos (PWM) nos permite generar salidas analógicas fácilmente en sistemas digitales. Por ejemplo, para mover un motor de corriente continua a diferentes velocidades o modificar el brillo de un LED

En este cuaderno técnico veremos cómo implementar un PWM sencillo, de frecuencia aproximada. Nos servirá para generar una salida analógica en sistemas donde la frecuencia no es un parámetro crítico. Existen otros sistemas donde la frecuencia tiene que tener un valor exacto, como el control de un servo. Eso lo veremos en otro cuaderno.

## Parámetros del PWM

Las señales PWM son periódicas, y tienen dos parámetros: la frecuencia y la anchura del pulso.

El periodo lo denotamos por  $T$  (el inverso de la frecuencia) y la anchura del pulso por  $W$ . La frecuencia (periodo) es un parámetro fijo: siempre es el mismo durante toda el funcionamiento del circuito, mientras que la anchura es variable: en cada ciclo puede tener un valor diferente



En esta figura vemos tres ciclos. Todos con el mismo periodo  $T$ , pero la anchura cambia en cada uno de ellos:  $W1$ ,  $W2$ ,  $W3$ ...



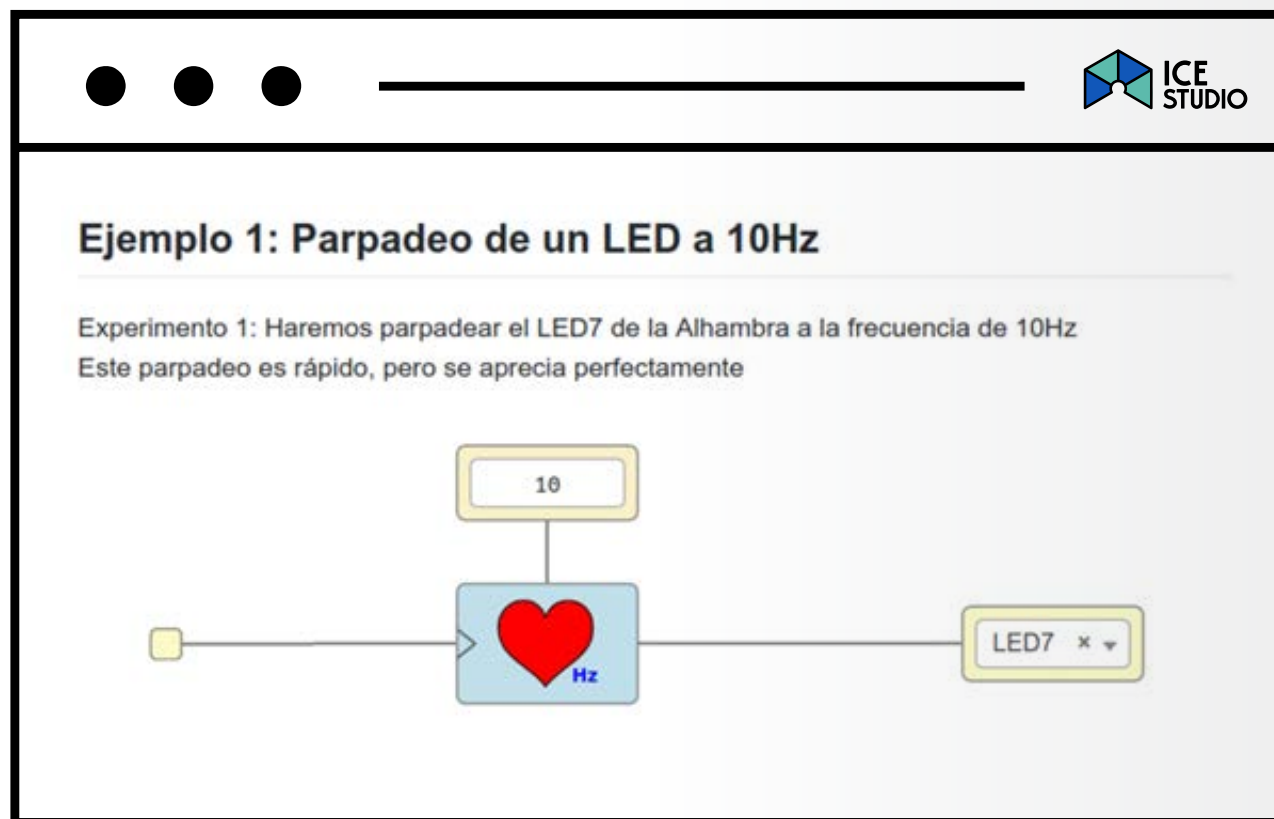
# Frecuencia

Según la aplicación, necesitamos seleccionar la frecuencia del PWM. Así, por ejemplo, para cambiar la intensidad de brillo de un LED basta con usar cualquier frecuencia mayor de 30Hz aproximadamente. A partir de ahí el ojo no nota el parpadeo.

Si usamos 10Hz, notaremos cómo el LED parpadea. Pero si usamos 60Hz NO. La frontera depende del ojo de cada uno, y del tipo de LED. Haremos algunos experimentos para entender esto mejor y practicar

## Experimento 1

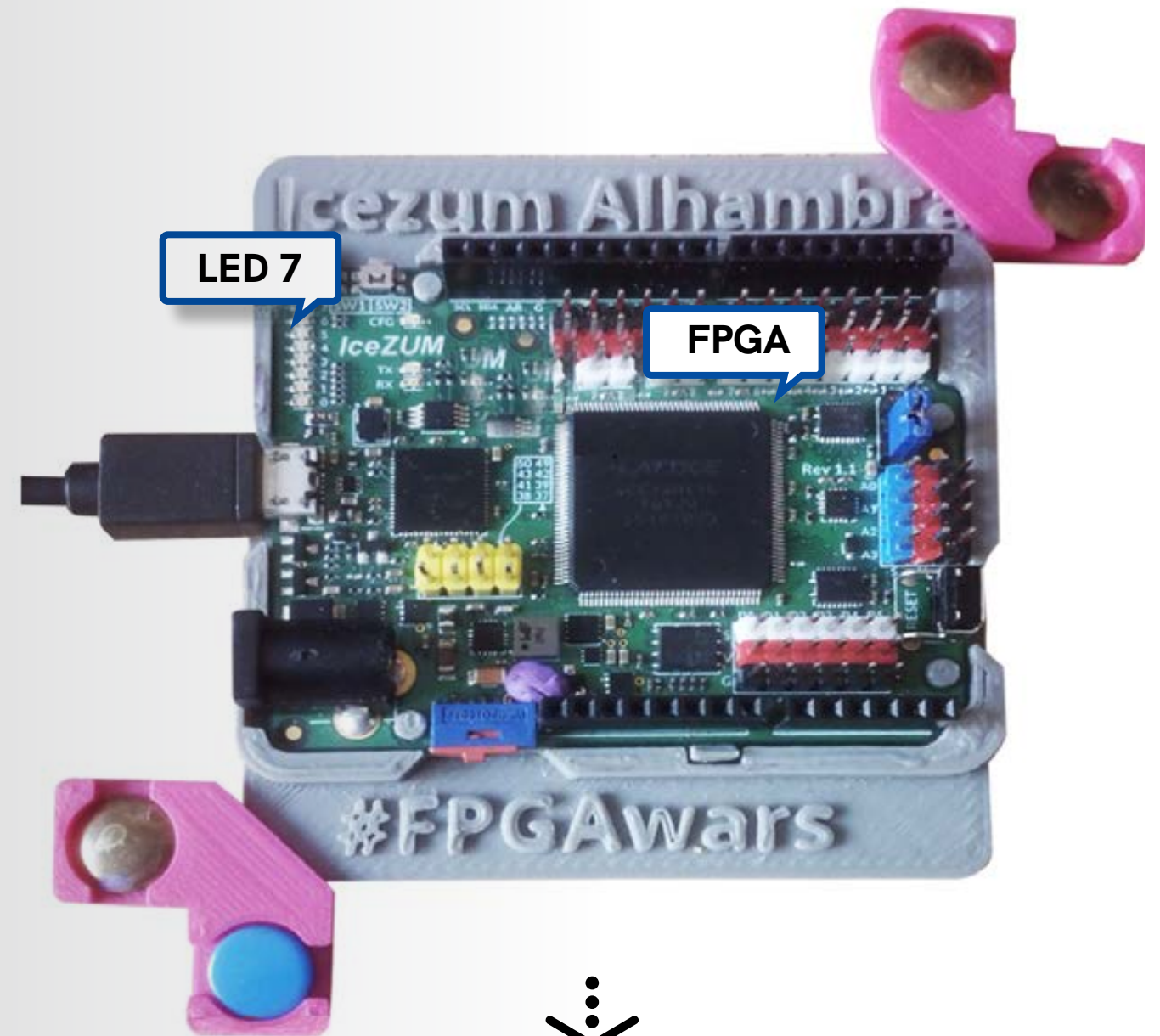
Parpadeo de un LED a 10Hz



Cargar este circuito en la placa Alhambra  
[Ej-01-LED-10Hz.ice](#)



El escenario está compuesto sólo por la placa Alhambra conectada al PC. Nada más.



Veremos cómo el LED7 parpadea, de una forma perfectamente apreciable.

# Experimento 2

Encontrar la frecuencia a la que no hay parpadeo

ICE STUDIO

### Ejemplo 2: Frecuencia a la que no se aprecia el parpadeo del LED

Experimento 2: Incrementar el valor de la frecuencia de parpadeo hasta que ya no se aprecie. Que no se aprecie no significa que no exista. El parpadeo está. Nuestro ojo **NO lo ve**, pero sí que lo podemos medir con un **analizador**

Fmin

40

Analizador

D0

LED7

Medir el parpadeo con el analizador lógico

Fmin es la frecuencia a la que ya **NO** se aprecia el parpadeo. Para cualquier otra frecuencia  $F > Fmin$  **NO** se ve el parpadeo

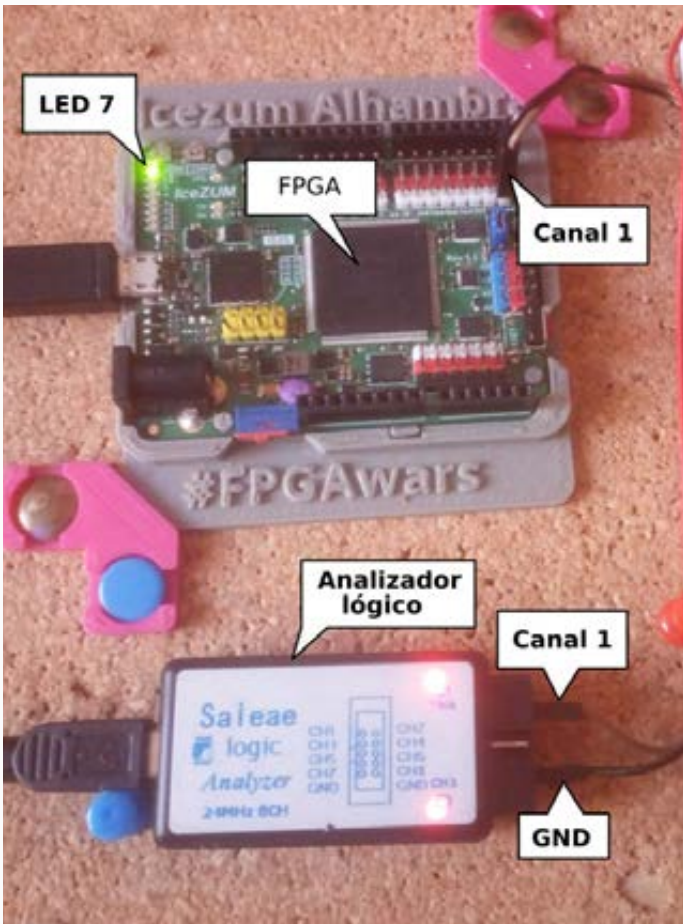
## El circuito queestoy usando

Partimos el ejemplo anterior, y aumentamos la frecuencia hasta que NO apreciemos el parpadeo. Probar con 20Hz, 25Hz, 30Hz, 40Hz... En mi caso, al llegar a 40Hz soy incapaz de apreciarlo (Bueno, en realidad a partir de 35Hz me es casi imposible apreciarlo).

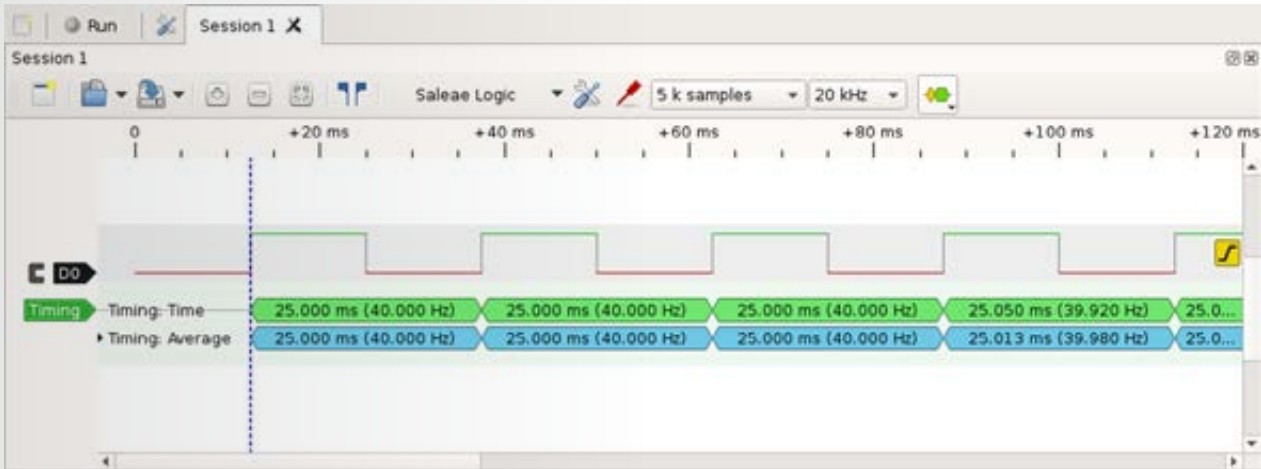
Sólo queda un cabo suelto. Asegurarnos que efectivamente la frecuencia de parpadeo del LED es la que le estamos indicando. Os lo podéis creer, o lo podéis medir. Para ello sólo hay que sacar la señal del parpadeo hacia el exterior (por el pin D0 por ejemplo) y usar el analizador compatible Saleae para medir.

“

Este es el nuevo escenario, con el analizador conectado al canal 1 para realizar la medición



Tomamos las medidas con el [PulseView](#)  
Este es el resultado obtenido:



Comprobamos que efectivamente la señal que llega al LED es de 40Hz



## Criterio de selección de frecuencia

A esta frecuencia mínima, a partir de la cual ya no se aprecia parpadeo la denominaremos  $F_{min}$ . El criterio para elegir la frecuencia de la señal PWM aplicada a un LED será:

La frecuencia  $F$  del PWM para modificar la intensidad de un LED sin apreciar parpadeo debe cumplir que  $F > F_{min}$

En nuestro caso  $F_{min} = 40\text{Hz}$ , así que cualquier frecuencia superior a  $40\text{Hz}$  nos servirá

# 40Hz



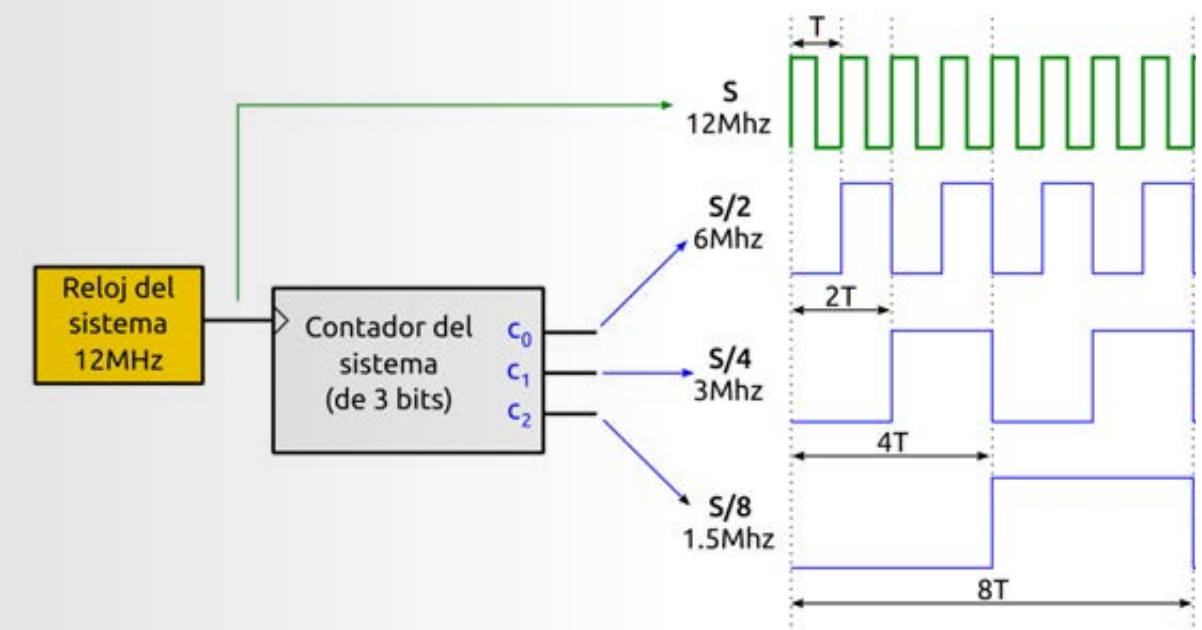
## Contadores

### Divisores de frecuencia

Los sistemas digitales están gobernados por un reloj, que llamamos el reloj del sistema, que tiene una frecuencia  $S$ . En las placas Alhambra esta frecuencia es de  $12\text{MHz}$ .

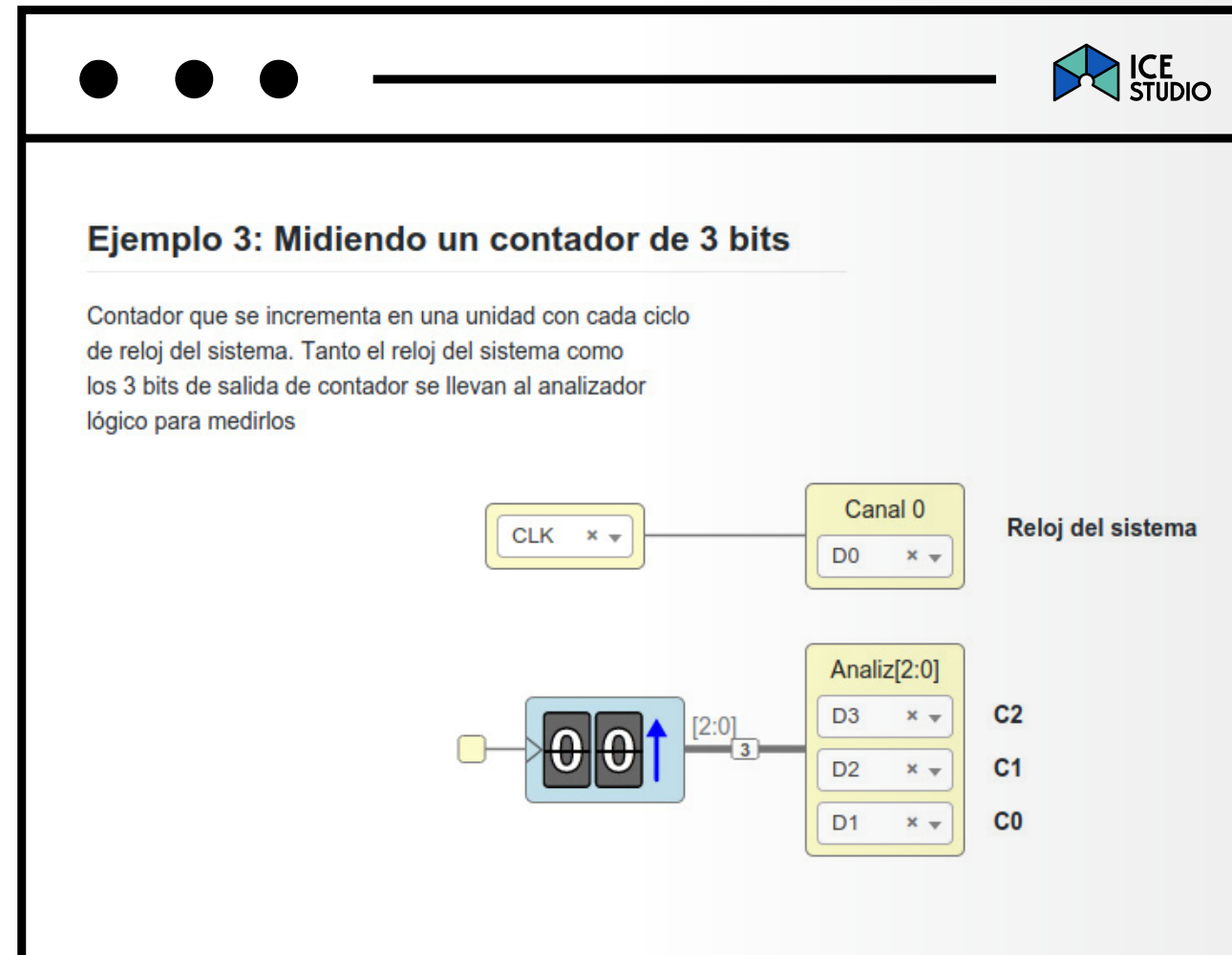
La forma más sencilla de obtener señales de menor frecuencia es usando contadores del sistema de  $N$  bits: son aquellos que se incrementan en una unidad con cada flanco de subida del reloj del sistema. Por el bit menos significativo del contador,  $c_0$ , se obtiene una señal de frecuencia  $S/2$ , por  $c_1$  de  $S/4$ , por  $c_2$  de  $S/8$ , etc.

Así, si tenemos un contador del sistema de 3 bits, las frecuencias que obtendremos serán  $S/2$ ,  $S/4$  y  $S/8$ . Como  $S$  es de  $12\text{MHz}$ , las frecuencias serán de  $6\text{MHz}$ ,  $3\text{MHz}$  y  $1.5\text{MHz}$



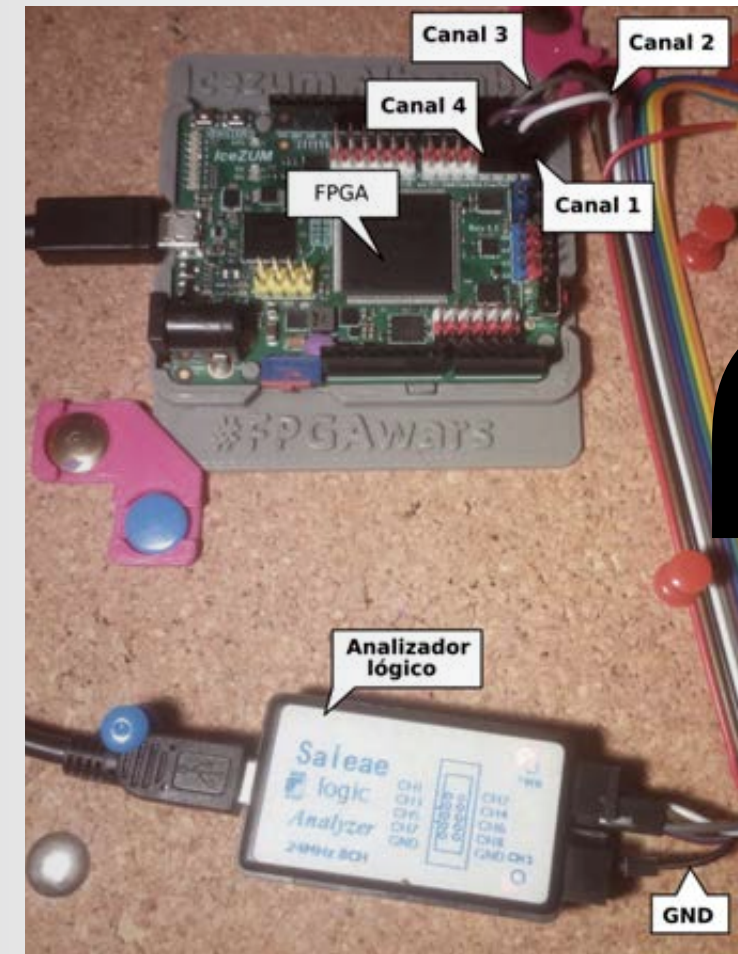
## Experimento 3

Midiendo un contador del sistema de 3 bits



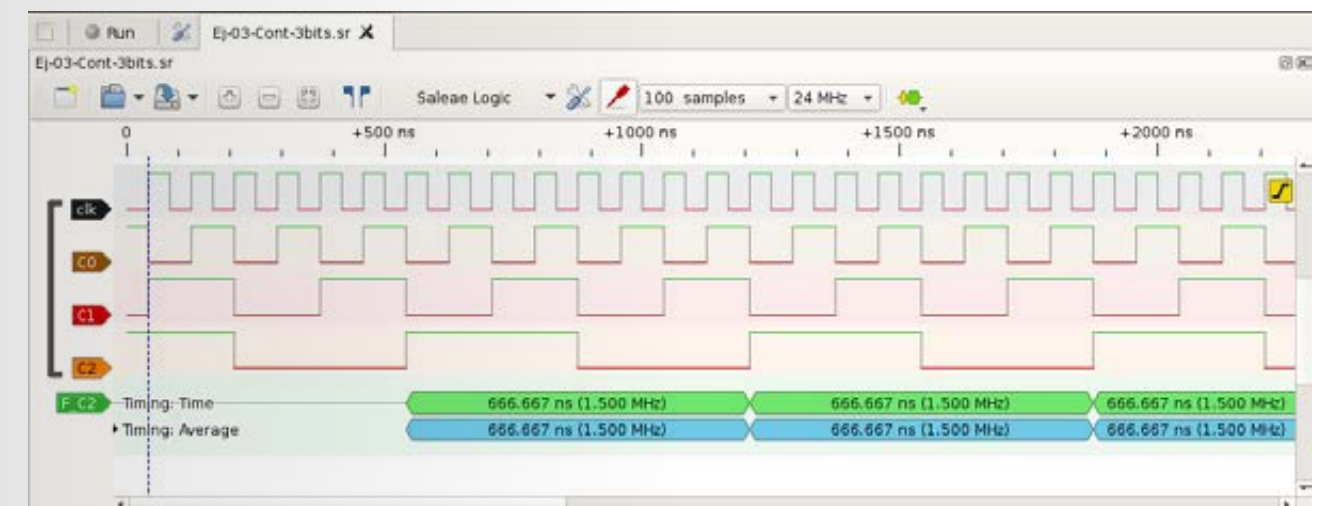
Comprobaremos lo dicho anteriormente experimentalmente:

Colocamos un contador del sistema de 3 bits con sus salidas conectadas al analizador lógico.



En el montaje colocamos el analizador, con cuatro canales conectados

Y realizamos la medición. comprobamos empíricamente que las señales del contador efectivamente son divisores de la señal de reloj del sistema. En F\_C2 podemos ver las medidas de la señal C2, que coincide con la frecuencia  $S/8$ , de 1.5Mhz



# Tabla de frecuencias

A esta frecuencia mínima, a partir de la cual ya no se aprecia parpadeo la denominaremos Fmin.

El criterio para elegir la frecuencia de la señal PWM aplicada a un LED será:

Bits (N)	Frecuencia	Descripción
0	12 MHZ	Reloj del sistema (S)
1	6 MHZ	S/2
2	3 MHZ	S/4
3	1.5 MHZ	S/8
4	750 KHZ	S/16
5	375 KHZ	S/32
6	187.5 KHZ	S/64
7	93.8 KHZ	S/128
8	46.9 KHZ	S/256
9	23.4 KHZ	S/512. Frecuencias NO audible
10	11.7 KHZ	S/1024. Frecuencia Audible
11	5.9 KHZ	S/2048
12	2.9 KHZ	S/4096
13	1.5 KHZ	S/8192
14	732 Hz	S/16384
15	366 Hz	S/32768
16	183 Hz	S/65536
17	92 Hz	S/131072
18	46 Hz	S/262144. Parpadeo NO aprecible
19	23 Hz	S/524288. Parpadeo apreciable
20	11 Hz	S/1048576
21	5.7 Hz	S/2097152
22	2.9 Hz	S/4194304
23	1.4 Hz	S/8388608


Hay dos valores fronteras. Para contadores con 9 ó menos bits se obtiene frecuencias mayores a 20KHz, que es el máximo audible por el oído humano. Cuando se controlan sistemas mecánicos, si se usan para el PWM frecuencias menores de 20Khz se podrían oír pitidos o zumbidos

Para contadores con 18 o menos bits, se tienen frecuencias mayores a 46Hz, por lo que si se aplican para el control del brillo de LEDs NO se apreciará el parpadeo

```
# Definir la unidad MHZ
MHZ = 1_000_000

# Reloj del sistema
S = 12 * MHZ

#-- Imprimir la lista de frecuencias
for i in range(24):
    F = round(S / (2 ** i))
    print(i,F)
```



➤ La tabla se genera fácilmente (en HZ) con este programa en python: `frec_list.py`

## Resumen criterio de selección de frecuencias

La regla es muy sencilla:

- Señales PWM para control de motores:  
Frecuencias superiores a 20KHz. Usar contadores de 9 ó menos bits
- Señales PWM para control del brillo de LEDs:  
Frecuencias superiores a 40Hz. Usar contadores de 18 o menos bits



# Estructura Unidad PWM

La estructura básica de la unidad PWM consta de 3 elementos:

## CONTADOR DE N BITS

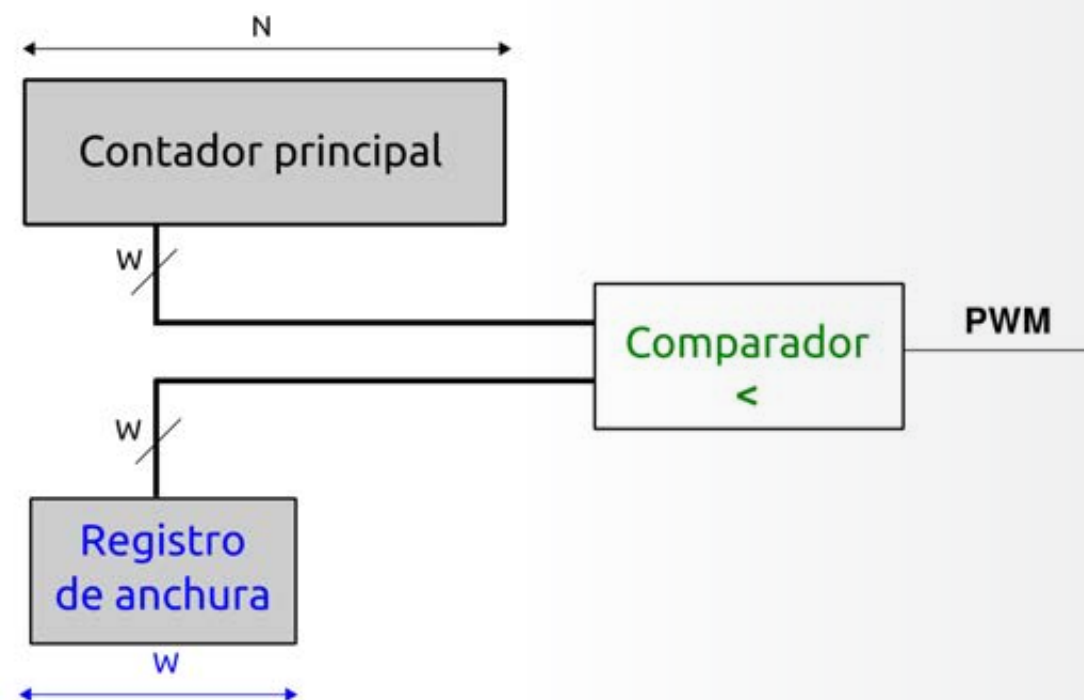
Donde N determina la frecuencia, y se elige en función de la tabla mostrada en apartado anterior

## REGISTRO DE ANCHURA, DE W BITS

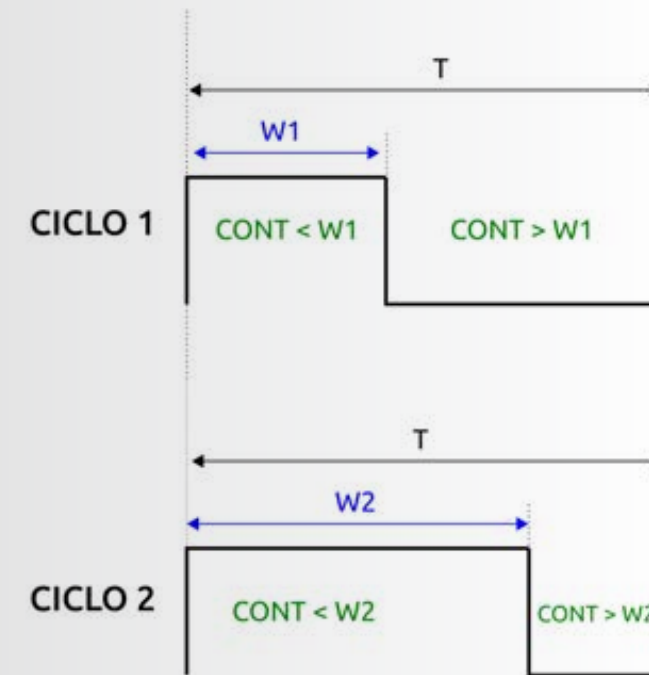
Contiene el valor de la anchura del pulso

## COMPARADOR DE W BITS

Donde N determina la frecuencia, y se elige en función de la tabla mostrada en apartado anterior.



La comparación se realiza entre el registro de anchura y los W bits más significativos del contador principal. Si representamos un único ciclo de PWM, observamos estas dos zonas: una donde el contador es menor al registro de anchura, y por tanto la salida está a 1, y otra donde el contador es mayor, cuya salida debe ser 0



En esta figura se representan dos ciclos del PWM, con anchuras de pulsos distintas: W1 y W2.

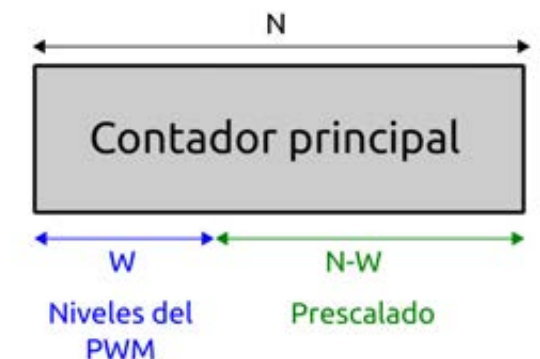
El contador está constantemente incrementándose, desde 0 (comienzo del ciclo) hasta su máximo valor (T). En el primer ciclo, cuando el contador es menor que W1, la señal está a 1. En El segundo ciclo es igual pero con otro valor: W2.

Esto hace que el periodo sea el mismo para ambos ciclos (T), pero que las anchuras sean diferentes

## Parámetro anchura del pulso

El parámetro W es el número de bits del registro de anchura. Este valor lo establecemos según los niveles que queramos para el PWM. Así por ejemplo, para el control de brillo de un LED podríamos establecer 256 niveles, o sólo 4. Es un valor que depende de la aplicación y lo determina el diseñador. Cuantos más niveles usemos, mayor será la resolución

Así, el contador principal está dividido en dos partes: la de mayor peso de W bits que determina el número de niveles, y la de menor peso, de N-W bits, que hace un prescalado. Nos permite tener el PWM con los mismos niveles pero a una frecuencia menor. En la mayoría de las veces tendremos que  $N = W$ , por lo que no se usa la parte de prescalado.



En general, en las aplicaciones de FPGA cuanto mayores sean las frecuencias, menos recursos se gastarán. Así que, si no hay otras restricciones, intentaremos siempre usar la máxima frecuencia, y que  $N = W$  (sin prescalado)

# Aplicación: Control del brillo de un LED

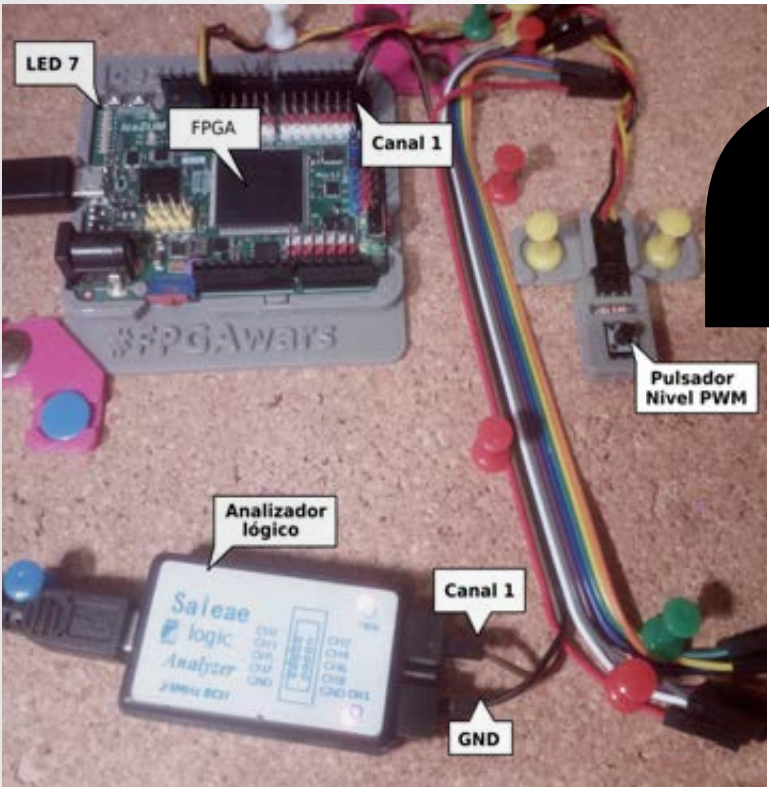
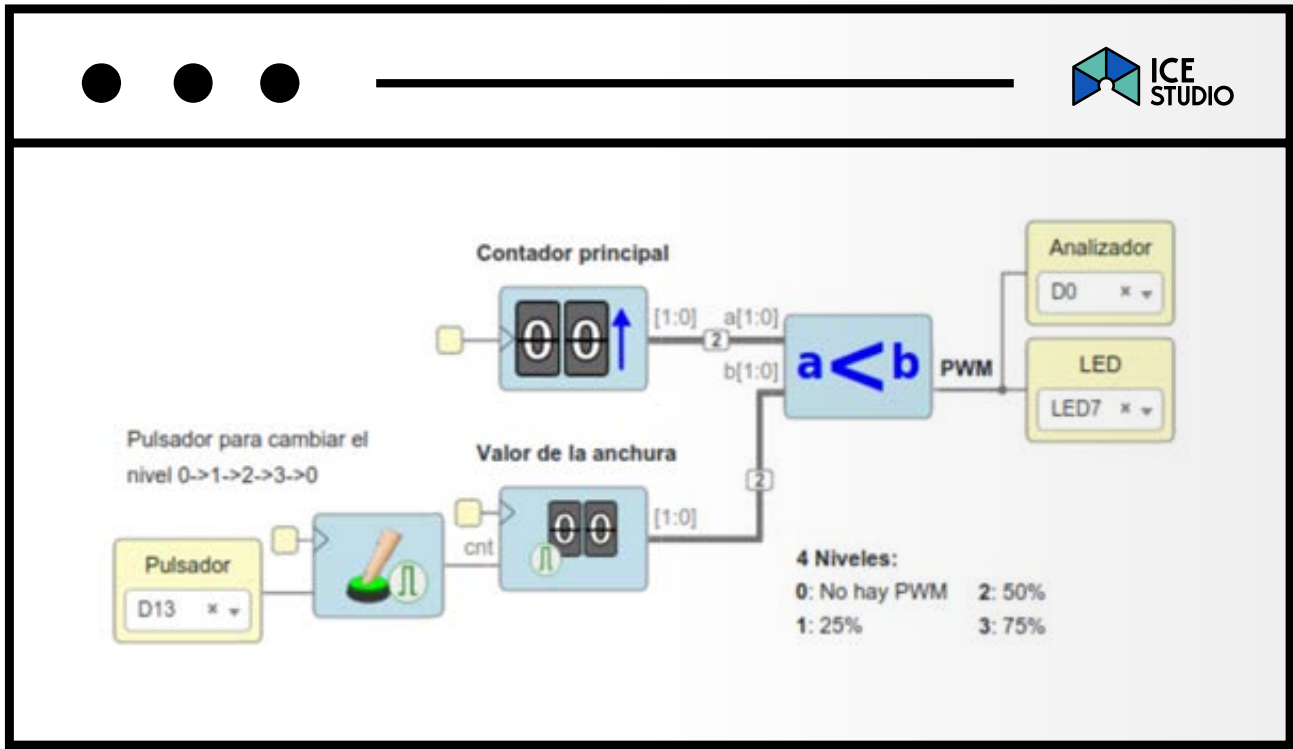
Como ejemplo de diseño de nuestra unidad de PWM, realizaremos una para controlar el brillo de un LED. Sabemos que la frecuencia deberá ser mayor de 40Hz para eliminar el parpadeo.

## Ejemplo 4

Cuatro niveles de brillo en el LED

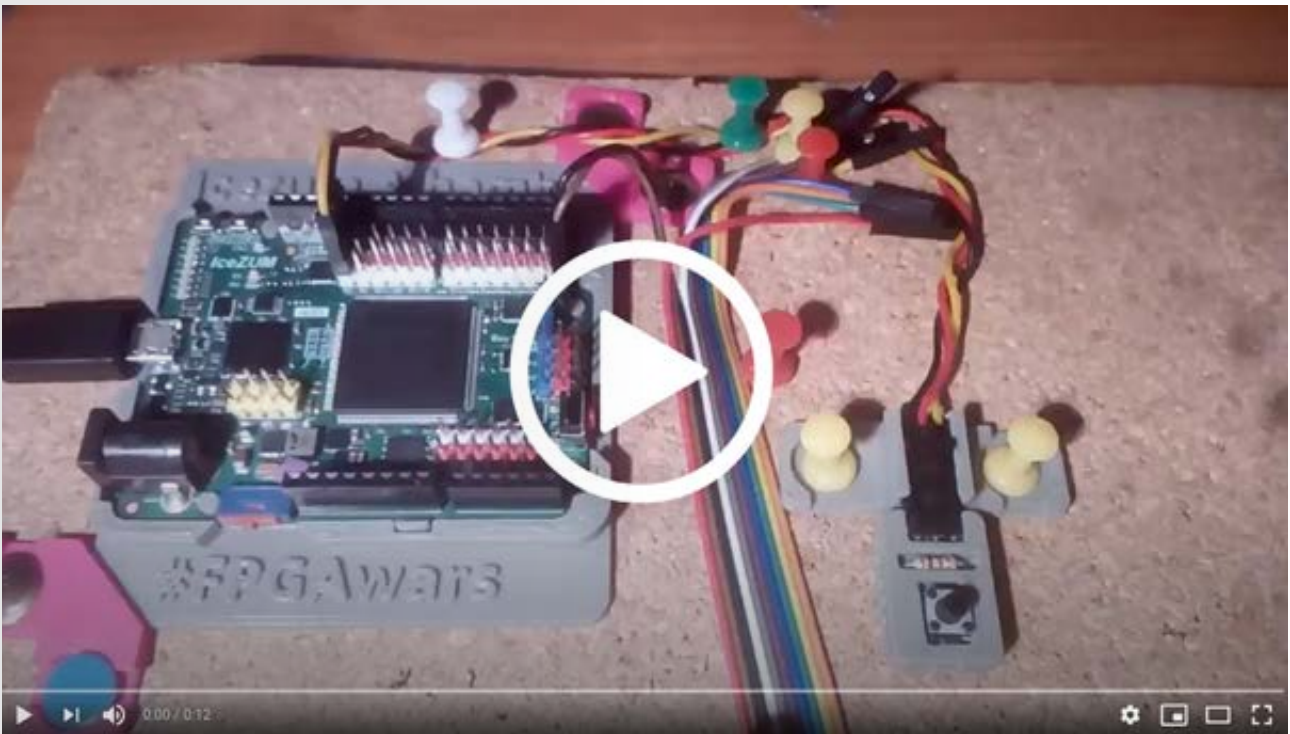
Empezaremos por el ejemplo más básico: implementar 4 niveles de brillo en el LED. Como queremos 4 niveles, tenemos que  $W = 2$ . Y como no hay restricciones adicionales, haremos que  $N = W = 2$ . Por tanto, según la tabla de frecuencias, tendremos una señal de PWM de 4 niveles a la máxima frecuencia: 3 MHz. El circuito consta del contador del sistema de 2 bits. Se incrementa en una unidad por cada ciclo de reloj del sistema (12MHz).

Para establecer el valor de la anchura utilizaremos otro contador de 2 bits, que se modificaremos con un pulsador externo. Cada vez que apretemos el pulsador incrementaremos el brillo del LED en un nivel. La señal de PWM se obtiene a la salida del comparador de 2 bits. Por un lado lo sacamos al LED7, que es el que queremos controlar su brillo. Por otro lado lo llevamos al pin D0 para medir la señal con el analizador lógico.



“

En el montaje tenemos el pulsador externo y el analizador lógico, con el canal 1 conectado



Lo cargamos y lo probamos. Al apretar el pulsador cambiamos la luminosidad del LED. En este vídeo se muestra en funcionamiento



En este pantallazo se muestran las medidas tomadas.

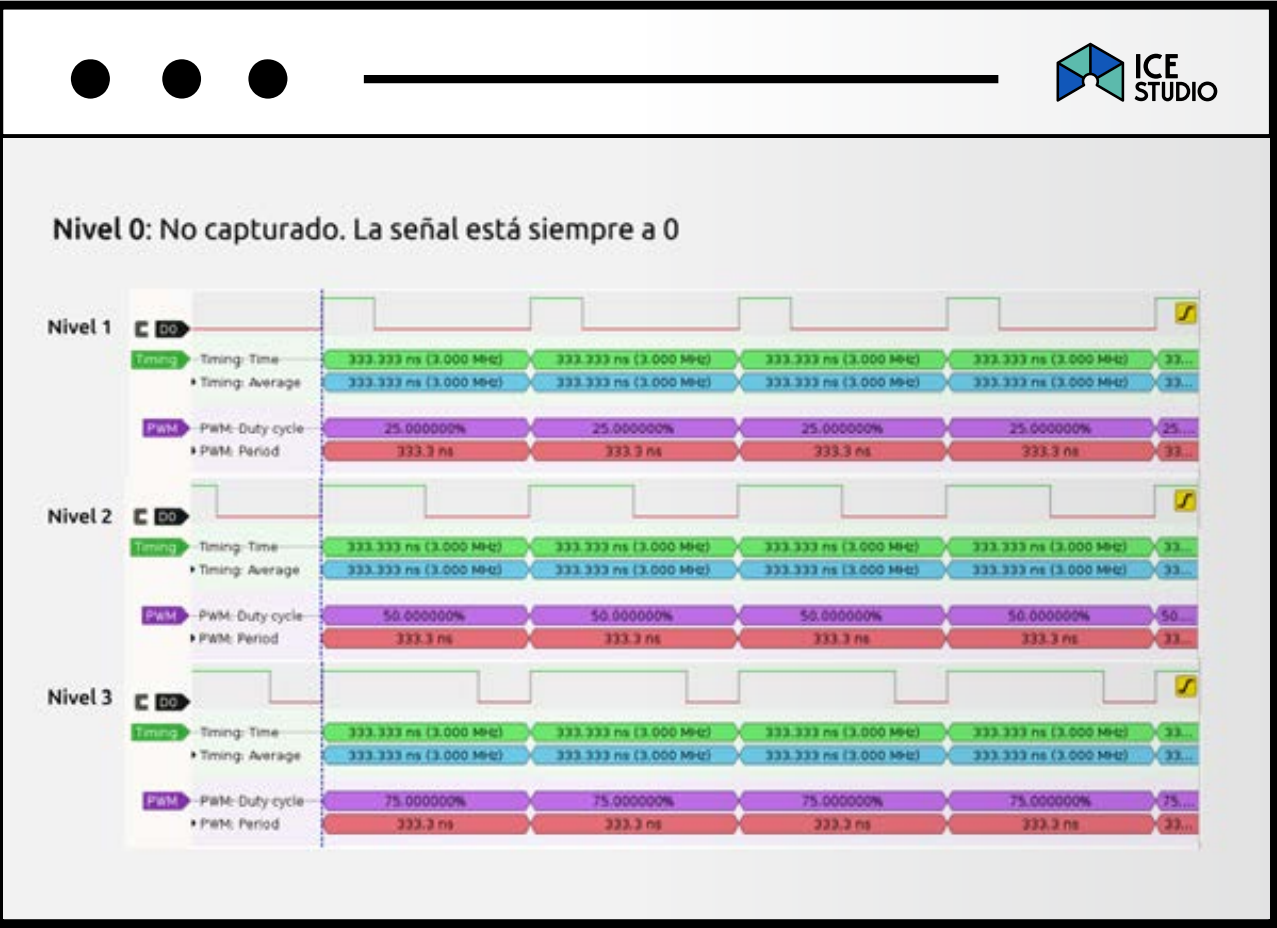


Se han tomado de forma individual: una medida para cada nivel y se ha agrupado en una única figura.

La medida del nivel 0 no se ha tomado, ya que la señal está siempre a 0.

La captura de la señal está a al derecha de la linea vertical azul.

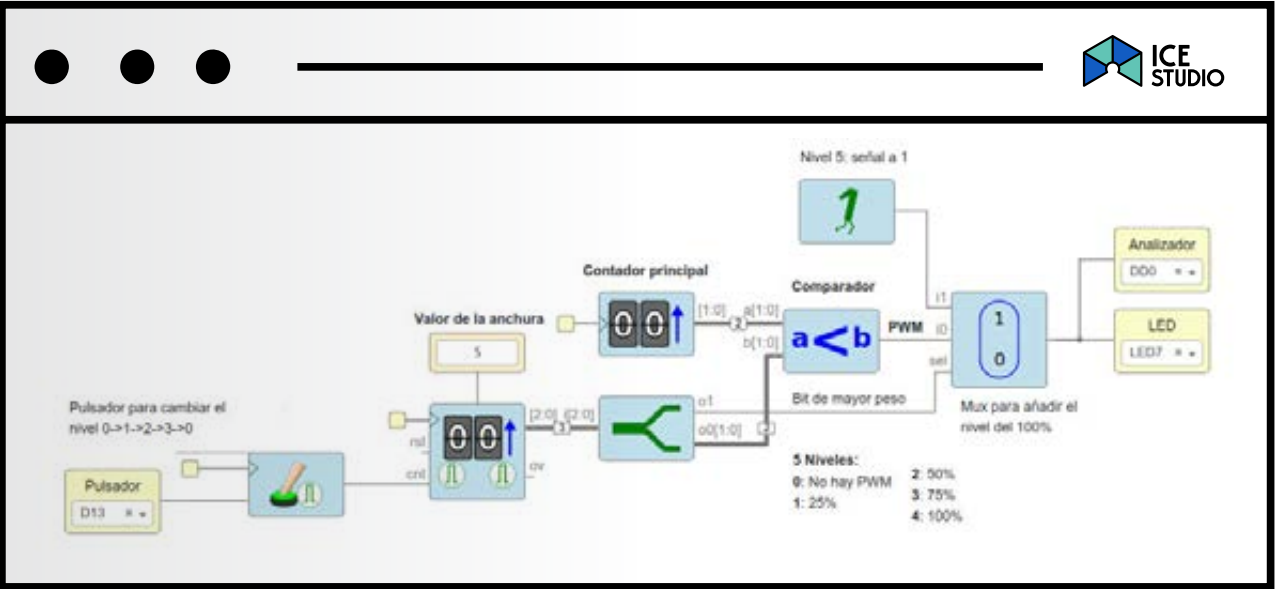
Comprobamos cómo el ancho es 0, 25%, 50% y 75%, para los niveles 0, 1, 2 y 3. También comprobamos que la frecuencia es fija: 3MHz



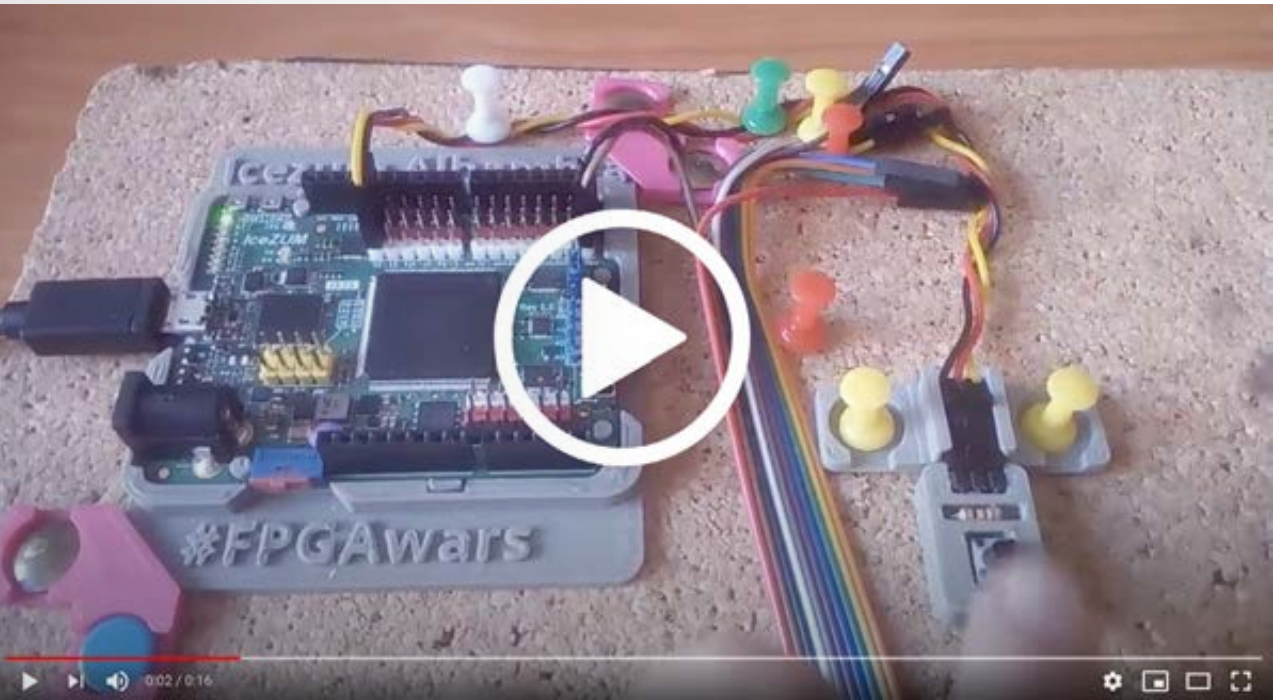
# Ejemplo 5

4+1 niveles de brillo en el LED

En el ejemplo anterior, el caso del brillo máximo, el 100%, NO está contemplado. Si lo queremos incluir, tenemos que añadir ese caso por separado. Es el caso en el la señal de pwm está siempre a 1. Hay que usar un contador módulo 5, y usar el bit de mayor peso para conmutar un multiplexor entre la señal PWM y la señal constante 1. Este es el circuito:



Lo cargamos y lo probamos. Ahora hay 4 niveles donde el LED está iluminado (25%, 50%, 75%, 100%) y uno en el que está apagado



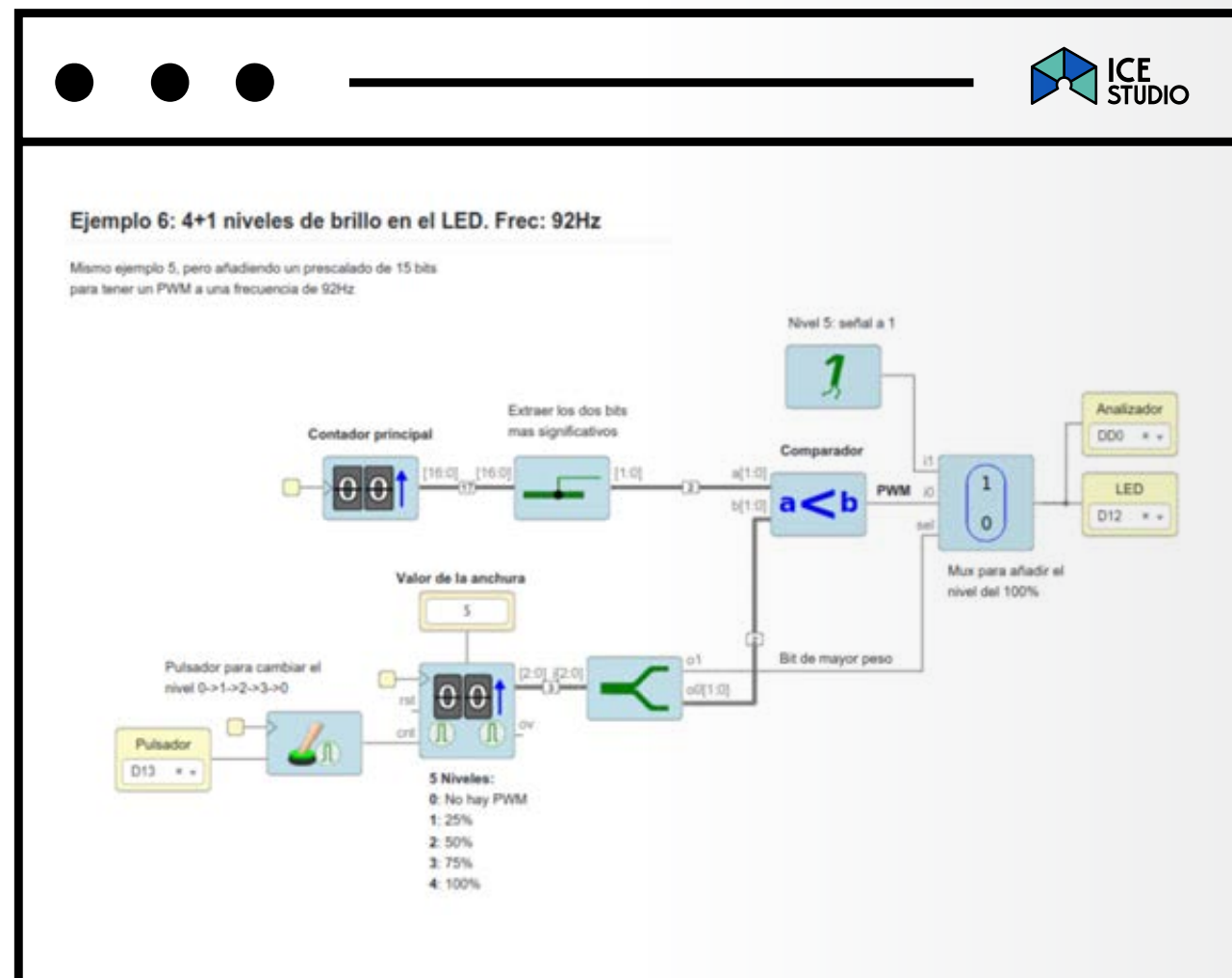


## Ejemplo 6

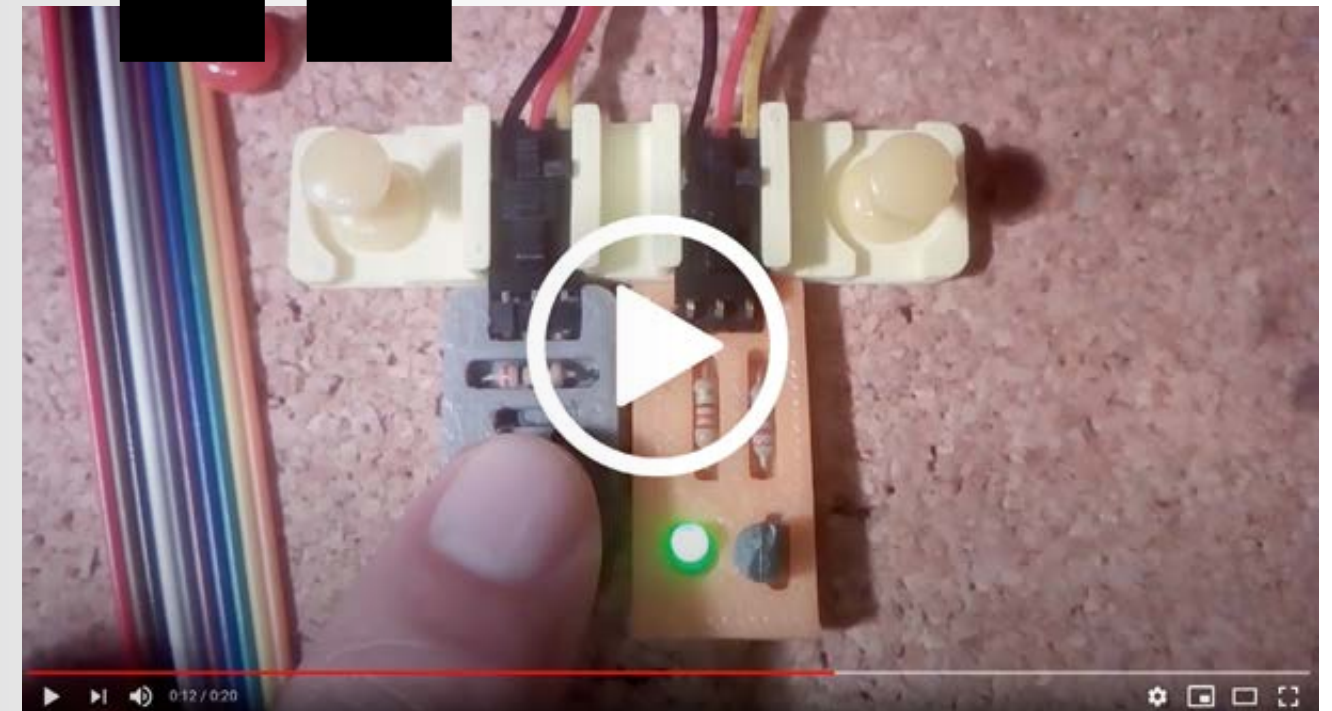
4+1 niveles de brillo con prescalado de 15 bits

En el ejemplo 5, la frecuencia del PWM es de 3 MHz. ¿Cómo podemos modificarlo para que funcione a otra frecuencia? Elegimos de la tabla de frecuencia la nueva frecuencia.

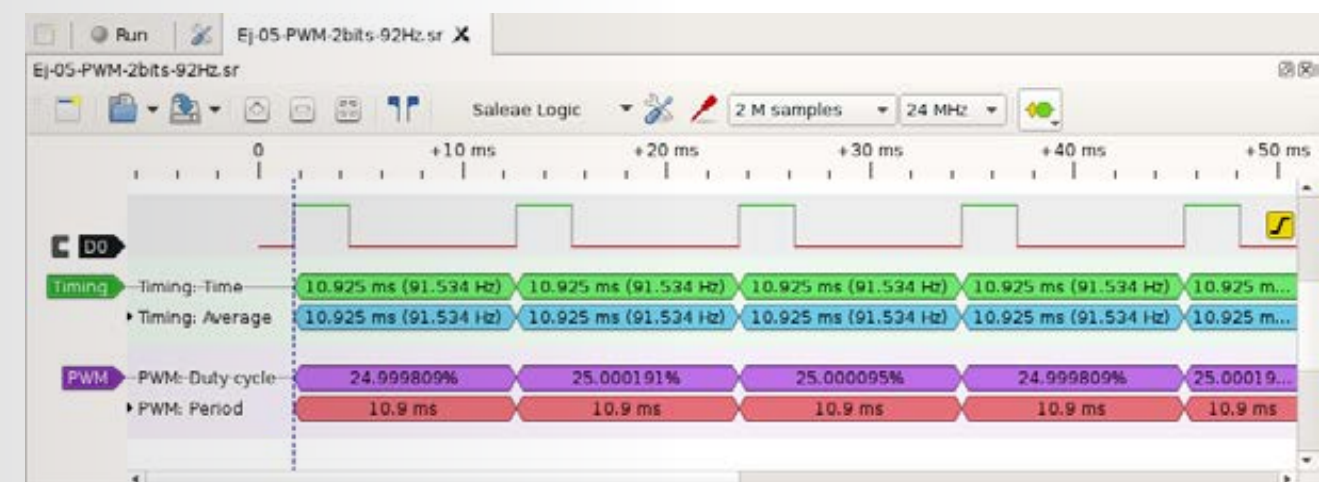
Por ejemplo 92Hz. Tenemos que ampliar el contador principal para que ahora tenga 17 bits: los dos bits más significativos son para determinar el nivel de brillo, y los 15 restantes para el prescalado. El circuito es el siguiente:



El escenario es el mismo, pero ahora usando un LED externo. Lo cargamos y lo probamos



Al realizar la medición, comprobamos que la frecuencia es de 91.53Hz (En la tabla de frecuencias es 92Hz, porque se ha redondeado al hz). El circuito hace lo mismo que en el ejemplo anterior, pero a una frecuencia menor



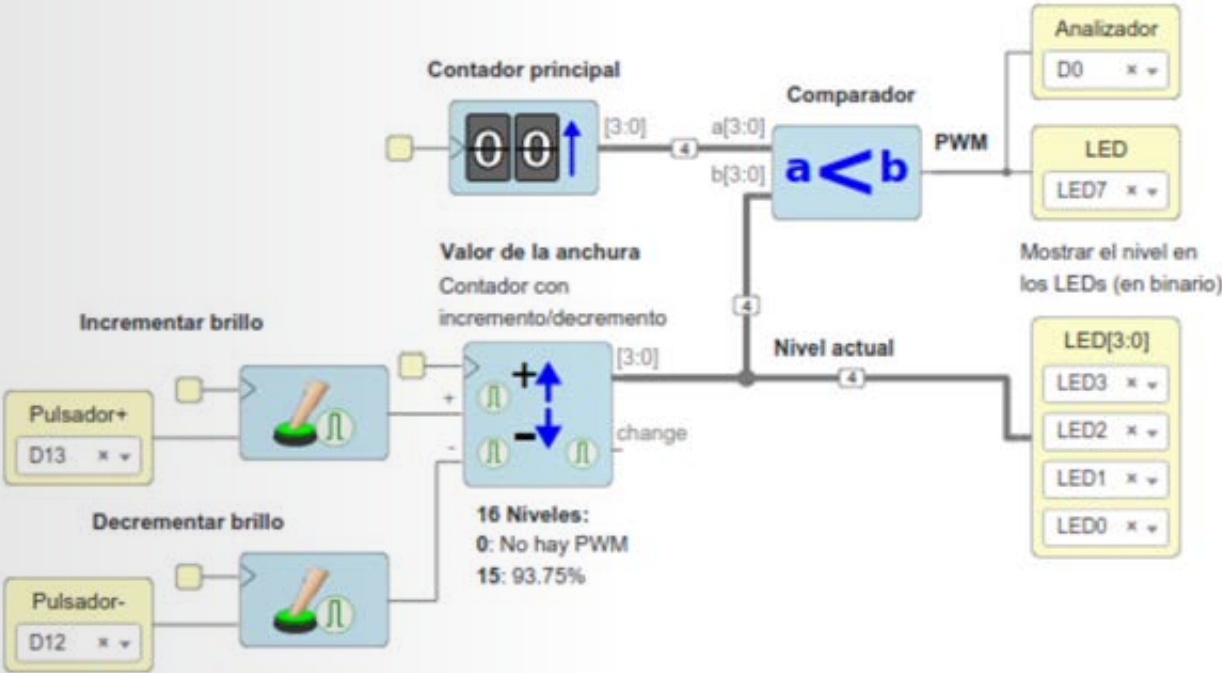
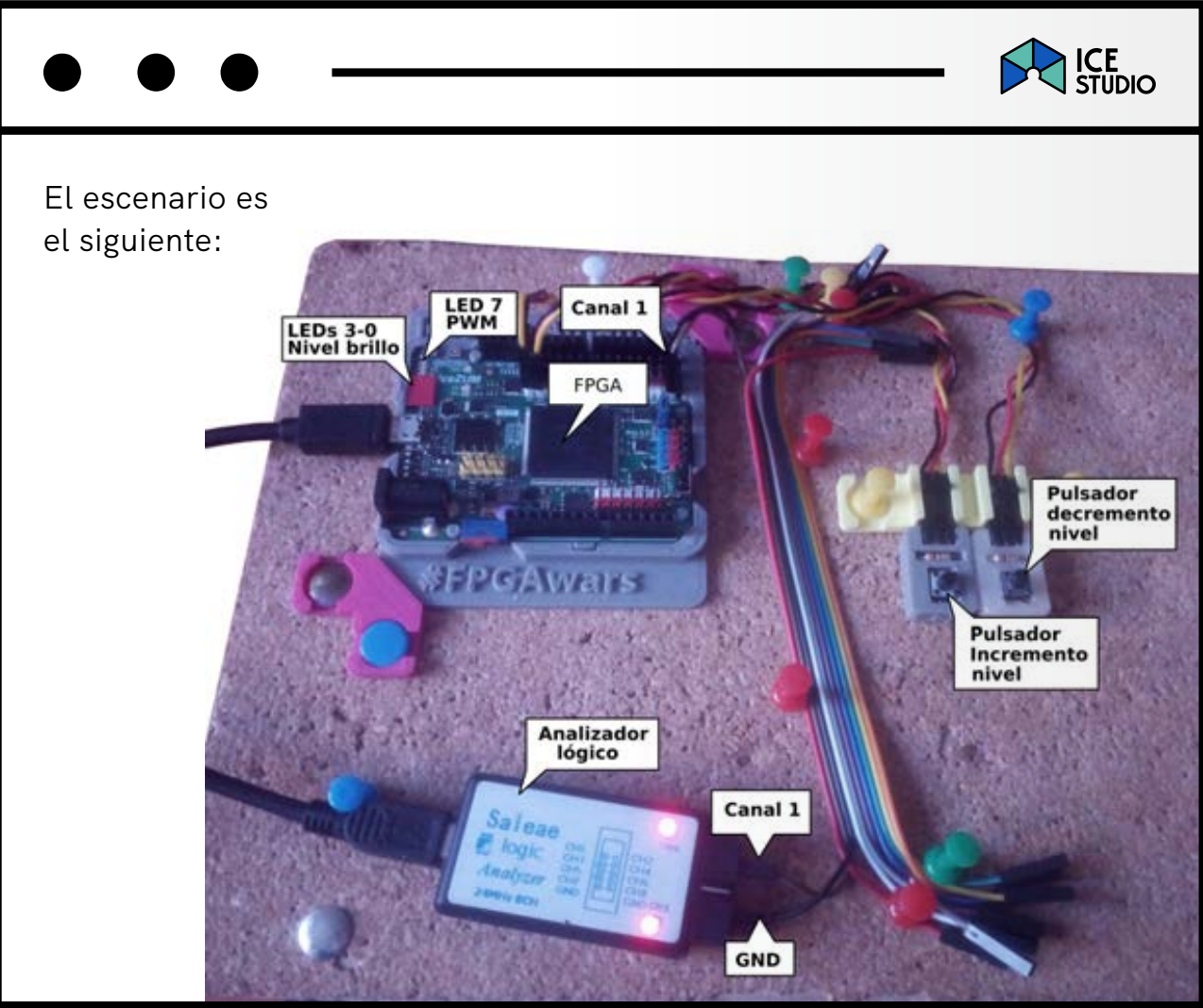


# Ejemplo 7

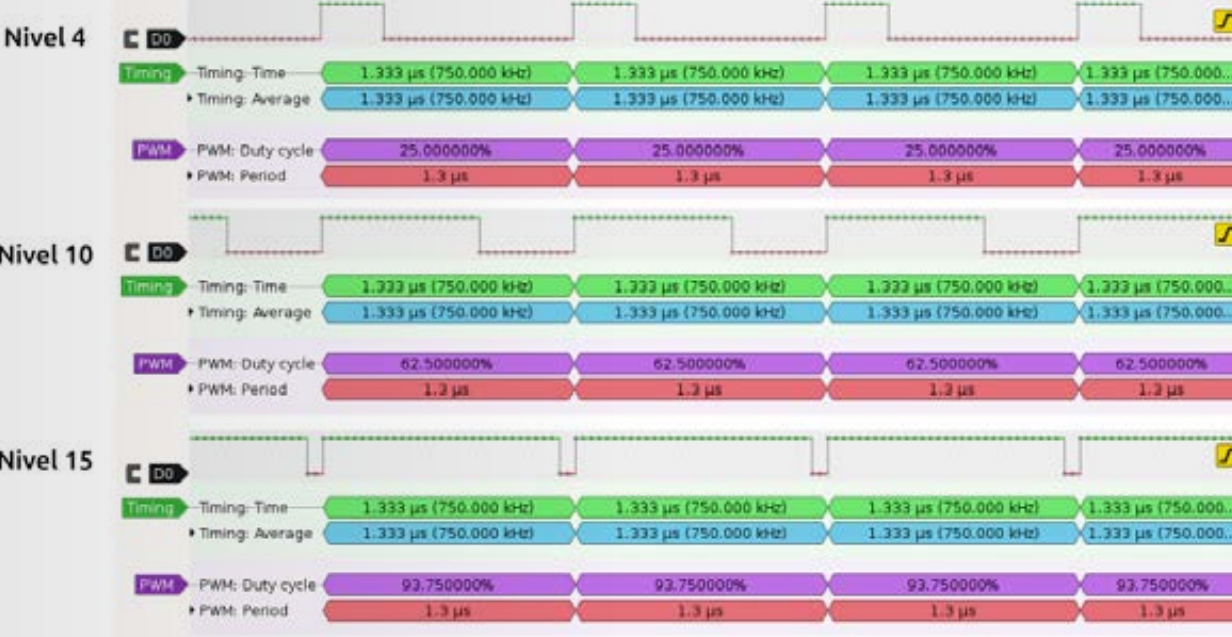
16 niveles de brillo, sin prescalado

Para aumentar el número de niveles, basta con usar un contador principal de mayor número de bits. Así, para tener 16 niveles de brillo, usamos un contador de 4 bits. No usaremos bits adicionales de prescalado, por lo que la frecuencia de este PWM será de 750Khz, según la tabla de frecuencias

Usamos dos pulsadores, uno para incrementar el brillo y otro para disminuirlo. Se usa un contador con incremento/decremento, que NO permite rebasar los límites: cuando se llega al valor máximo (15) no se puede seguir incrementándolo. Y al llegar al valor mínimo (0) no se puede decrementar más. El escenario es el siguiente:



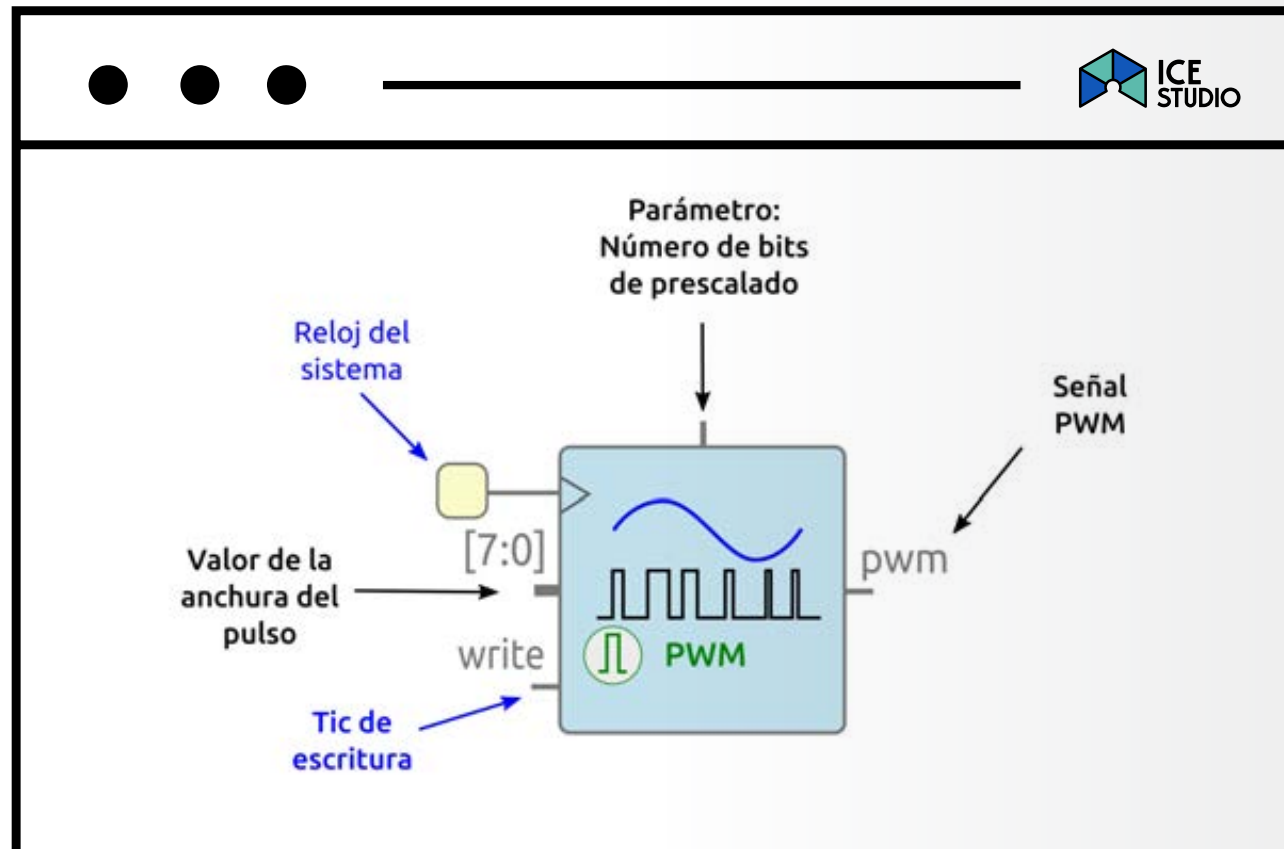
Lo cargamos y lo probamos. Con el pulsador izquierdo incrementamos el nivel en una unidad, y con el derecho lo decrementamos. En el vídeo no se aprecian bien los 16 niveles, pero en real sí se ve la diferencia, especialmente en los niveles bajos. El nivel actual se muestra en binario en los LEDs de menor peso



En este pantallazo se muestran las medidas tomadas para los niveles 4 (25%), 10 (62.5%) y 15 (93.75%). Se han tomado por separado y se han unido en la figura. Se comprueba que la frecuencia es de 750Khz en todos los casos

# Bloque de PWM

Las unidades de PWM se encuentra en la colección Jedi, en el menú Varios/PWM. Como parámetro se especifica el número bits de prescalado, que por defecto es de 0 (sin prescalado, máxima frecuencia)

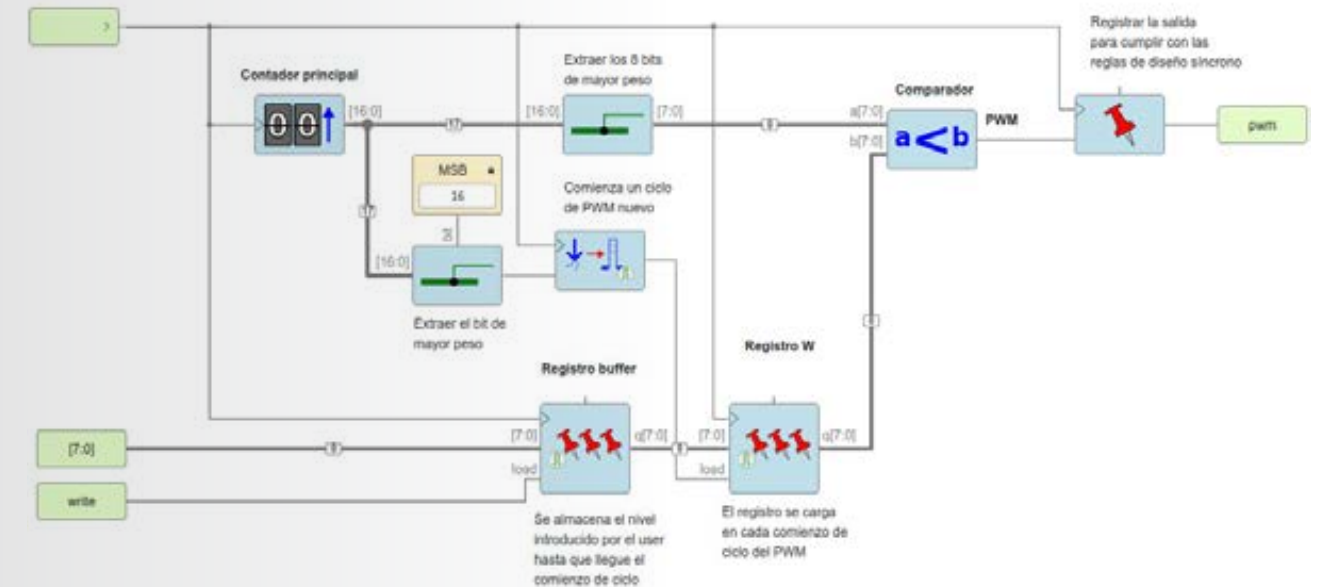



Este bloque funciona como un registro: Se le introduce el valor de la anchura del pulso en el bus de entrada y se emite un tic de escritura en write para que lo capture. Por la salida se obtiene la señal PWM. Existen diferentes unidades de pwm según el número de niveles. El que se muestra en la figura es de 8bits: 256 niveles

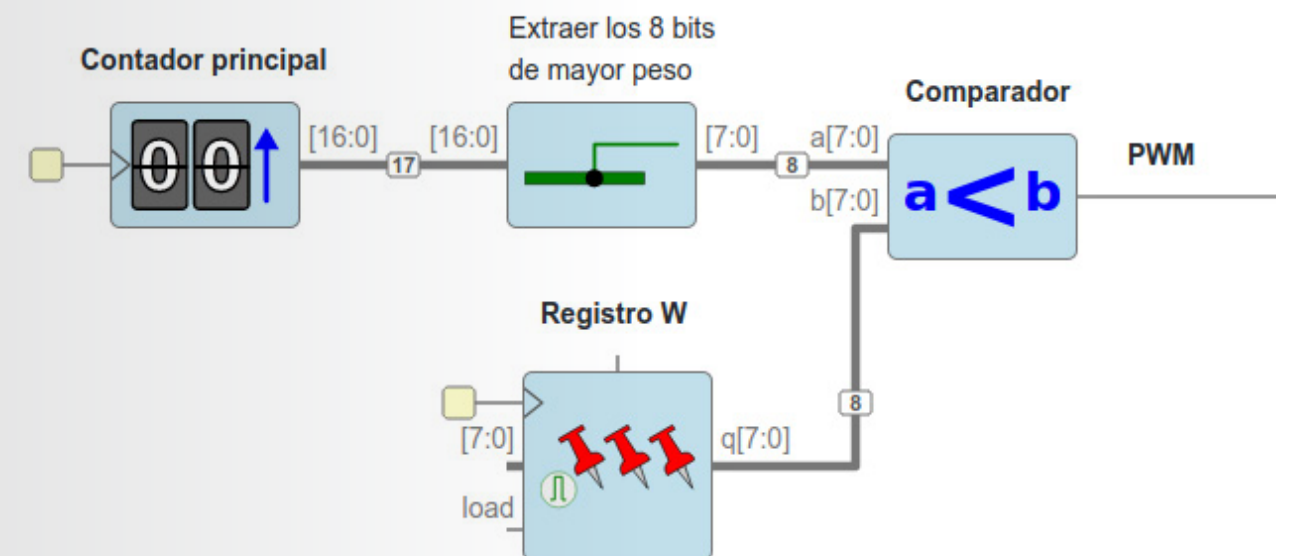
La frecuencia de funcionamiento es la máxima por defecto. Para calcular la frecuencia que tiene basta con sumar los bits de prescalado y los bits de la anchura, y acudir a la tabla de frecuencias. Cambiando los bits de prescalado podemos bajar la frecuencia

## Funcionamiento interno por bloques

Como ejemplo de funcionamiento veremos cómo está hecho el bloque pwm de 8 bits y frecuencia de 92Hz. Este bloque, compuesto por otros bloques, se encuentra en el menú Varios/PWM/08-bits/blocks/pwm-92Hz



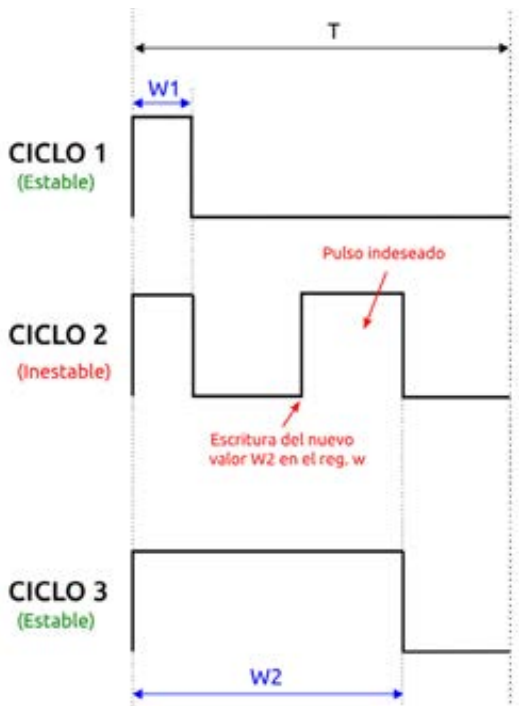
 Como ejemplo de funcionamiento veremos cómo está hecho el bloque pwm de 8 bits y frecuencia de 92Hz. Este bloque Veremos su funcionamiento detenidamente. Como es una unidad de 92Hz, necesitamos un contador principal de 17 bits. El esquema es como el general que ya conocemos: el contador principal, el registro de anchura w, de 8 bits y el comparador de 8 bits. Los 8 bits más significativos del contador los extraemos (usando el bloque extractor) y los comparamos con los 8 bits del registro w: e, compuesto por otros bloques, se encuentra en el menú Varios/PWM/08-bits/blocks/pwm-92Hz





El valor de la anchura se debe introducir en el registro w, sin embargo, NO se puede hacer en cualquier momento. Si queremos tener una señal PWM correcta, en la que la frecuencia se mantiene fija, el registro w sólo se puede actualizar al comienzo de cada ciclo pwm. Si el usuario puede escribir en cualquier momento la nueva anchura, aparecerán pulsos indeseados

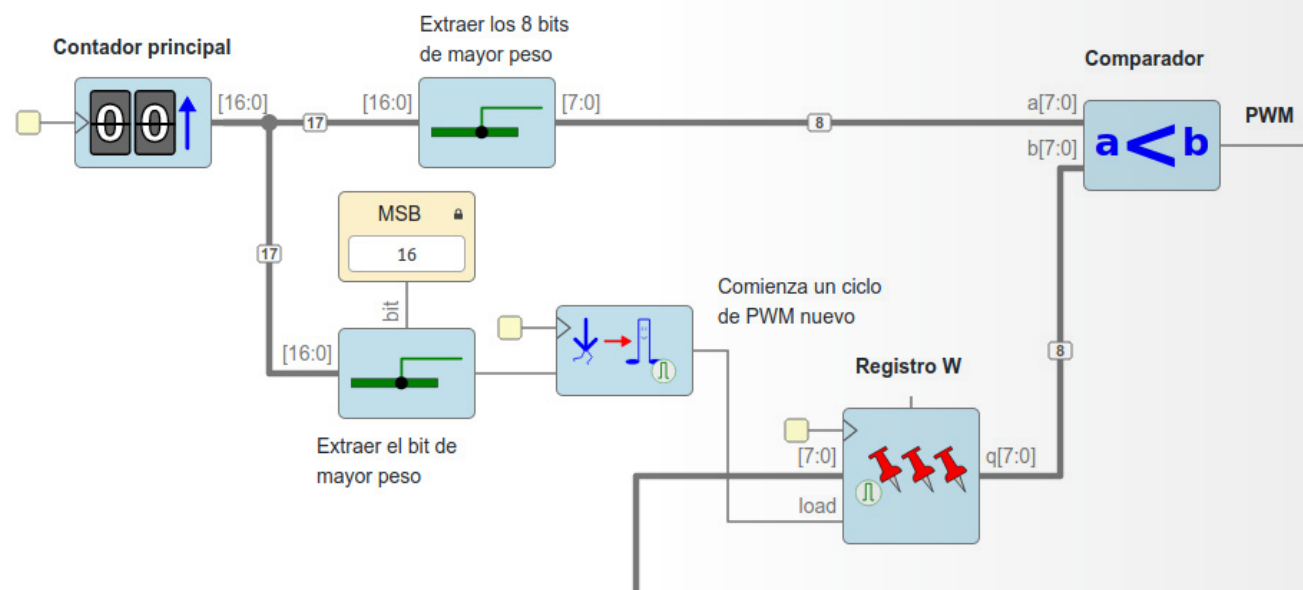
Este efecto se muestra en la siguiente figura. Partimos de una anchura W1 que es estable (ciclo 1). Queremos escribir una nueva anchura, w2, como la que aparece en el ciclo 3, que es estable. La transición se realiza en mitad del ciclo 2. Como el valor que estaba escrito era el w1, el ciclo comienza igual que el ciclo anterior: con un pulso de anchura w1.



Sin embargo, al escribirse el nuevo valor w2, se cumple que el contador en ese instante es menor que w2, por lo que la señal de pwm se pone a 1: aparece un segundo pulso NO deseado.

La frecuencia en el ciclo 2 ha cambiado. En el ciclo 3 la situación vuelve a la normalidad

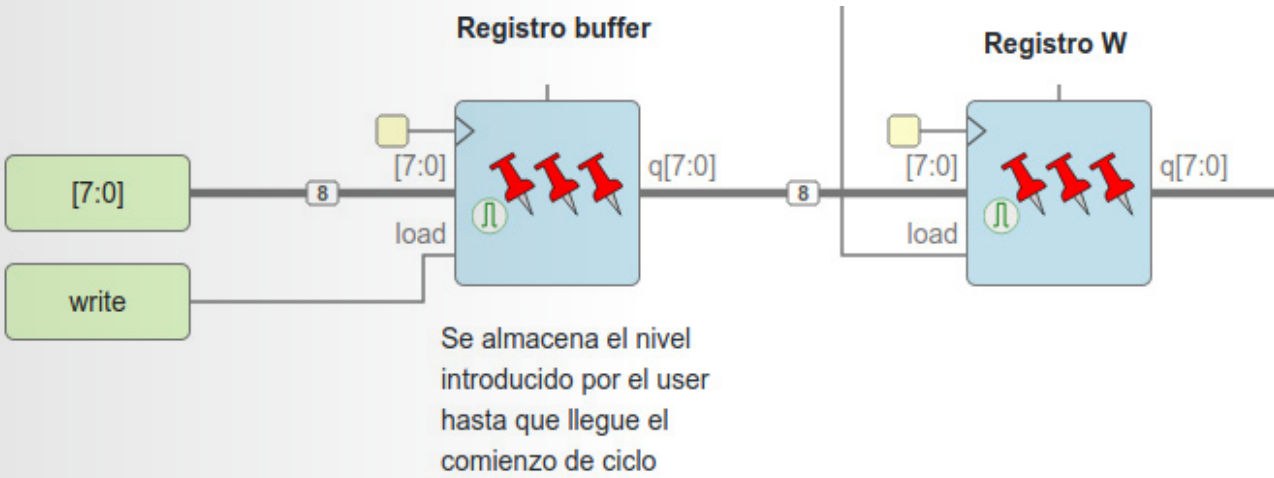
La manera de solucionarlo es haciendo que el registro w sólo se pueda actualizar al comienzo de cada ciclo.



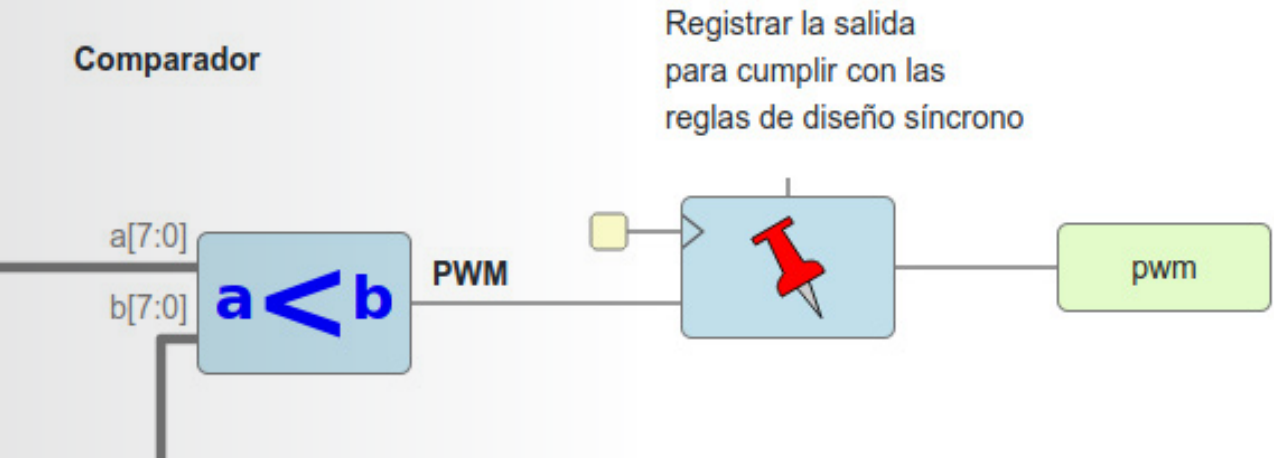
## Funcionamiento interno por bloques

Para detectar cuando empieza un ciclo nuevo, extraemos el bit de mayor peso del contador (bit 16) y emitimos un tic en el flanco de bajada: al pasar de 1 a 0. Eso sólo ocurre cuando ha transcurrido un periodo completo. Ese tic es el que usamos para capturar el nuevo valor del registro w. Y así garantizamos que no aparezcan pulsos no deseados

¿Y cómo hacemos ahora para que el usuario pueda escribir valores en el registro w? Usamos un registro buffer. Es un registro temporal en el que el usuario escribe el valor de la anchura cuando quiera. Este valor del buffer se transfiere al registro w en el comienzo de cada ciclo



Para terminar nuestra unidad, sólo necesitamos registrar la salida del pwm, usando un biestable del sistema, para cumplir con las normas del diseño síncrono. Al colocar un registro, eliminamos los glitches que pudiesen aparecer en la señal, antes de sacarla por el pin:



## Código verilog

Las unidades PWM que está fuera del directorio blocks están implementadas en Verilog, e incluyen el parámetro del prescalado. En los bloques de icestudio la entrada de la anchura es fija, pero en Verilog es genérica. El diseño es el mismo mostrado en el apartado anterior, pero descrito en verilog:

```
//-- Parámetro P: Número de bits del prescaler
//-- (P = 0 para no usar prescaler)

//-- Bits para el nivel
localparam N = 8;

//-- Contador principal
//-- Tamaño: Bits anchura + Prescaler (P)
localparam C = N + P;

reg [C-1:0] counter = 0;
always @(posedge clk)
    counter <= counter + 1;

//-- Detectar el comienzo de un ciclo nuevo:
//-- cuando hay un flanco de bajada en el bit de
//-- mayor peso (C-1)

reg q = 0;
always @(posedge clk)
    q <= counter[C-1];

//-- Cuando cycle_begin es 1, indica que comienza
//-- un nuevo ciclo
wire cycle_begin = q & ~counter[C-1];

//-- Registro W: Almacena la anchura actual
reg [N-1:0] reg_w = 0;

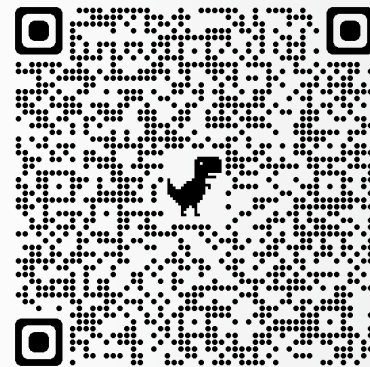
always @(posedge clk)
    //-- Se carga en cada nuevo ciclo de pwm
    if (cycle_begin)
        reg_w <= reg_buf;

//-- Registro buffer. Es donde se almacena la anchura
//-- introducida por el usuario mientras llega un
//-- nuevo ciclo de pwm, y se pueda cargar en el
//-- registro w
reg [N-1:0] reg_buf = 0;

always @(posedge clk)
    //-- Se actualiza cuando llega un dato nuevo
    if (write)
        reg_buf <= w;

//-- Salida del pwm: comparador
wire pwm_t = (counter[C-1:C-N] < w);

reg pwm = 0;
//-- Registrar la salida del pwm
always @(posedge clk)
    pwm <= pwm_t;
```



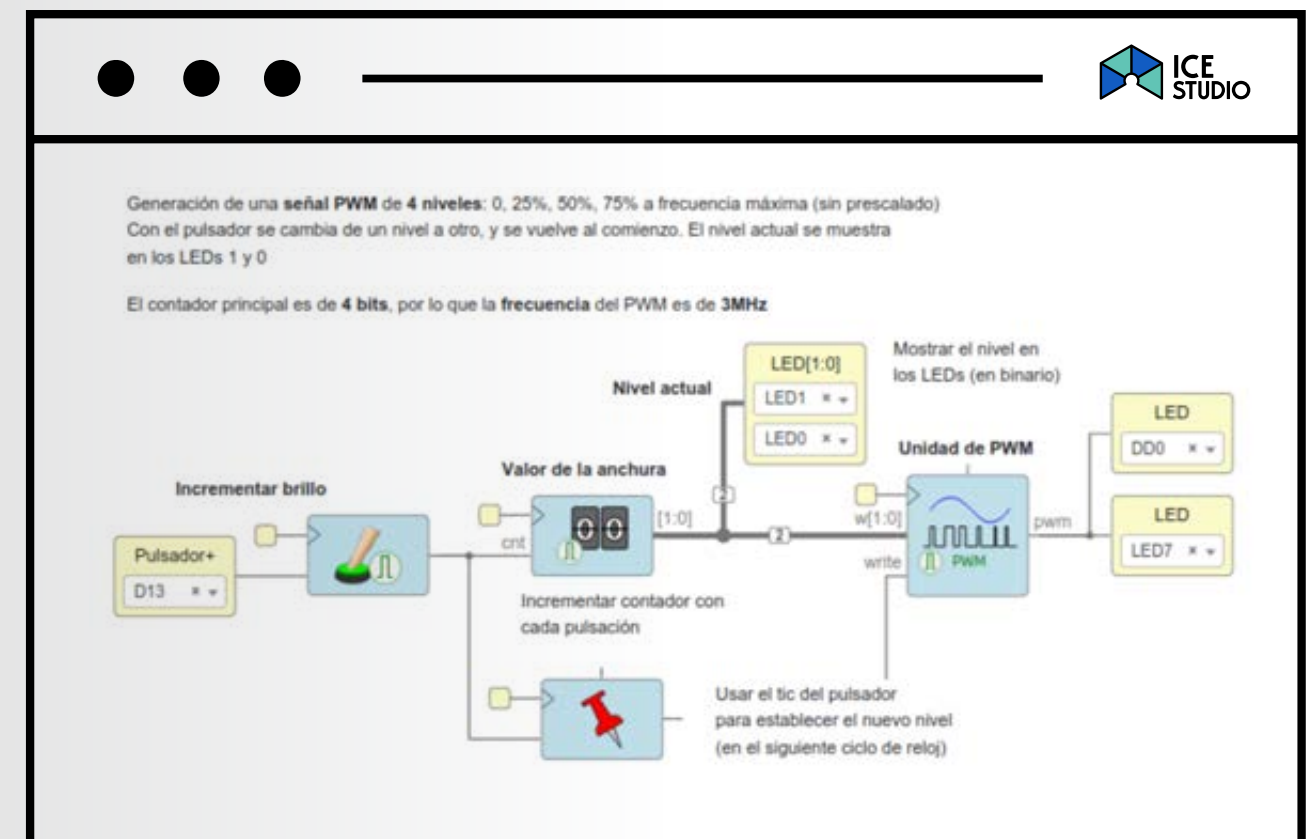
# Ejemplos de uso del bloque PWM

Ya hemos usado el PWM en varios ejemplo, pero aquí haremos ejemplos usando el bloque PWM. Todos estos ejemplos se encuentran en la colección Jedi.

## Ejemplo 1

Cuatro niveles de brillo en LED, con pulsador

Señal PWM de 4 niveles: 0, 25%, 50% y 75% y frecuencia de 3 MHz para controlar el brillo de un LED con el pulsador. El nivel de brillo actual se muestra en los LEDs 1 y 2 (2 bits). El LED controlado es el LED 7

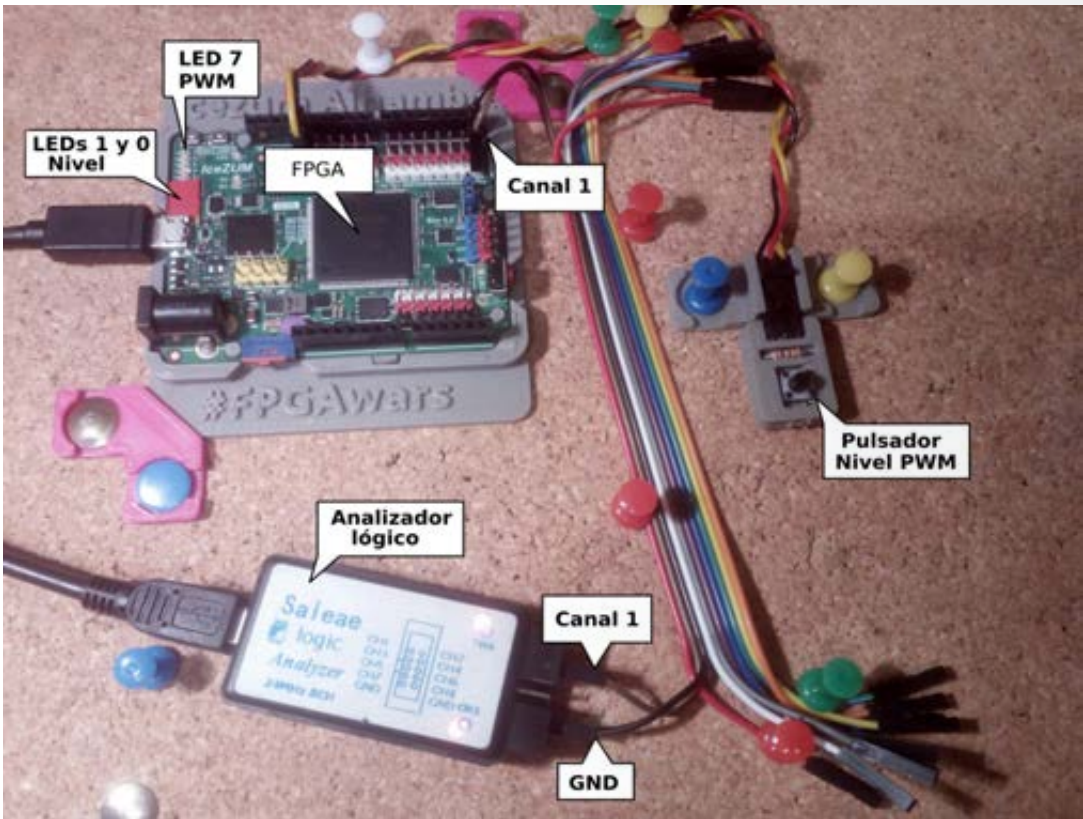


Descargar Fichero  
01-Brillo-LED-pulsador-2bits.ice



Escenario

- Elementos: Pulsador externo y analizador lógico para las mediciones

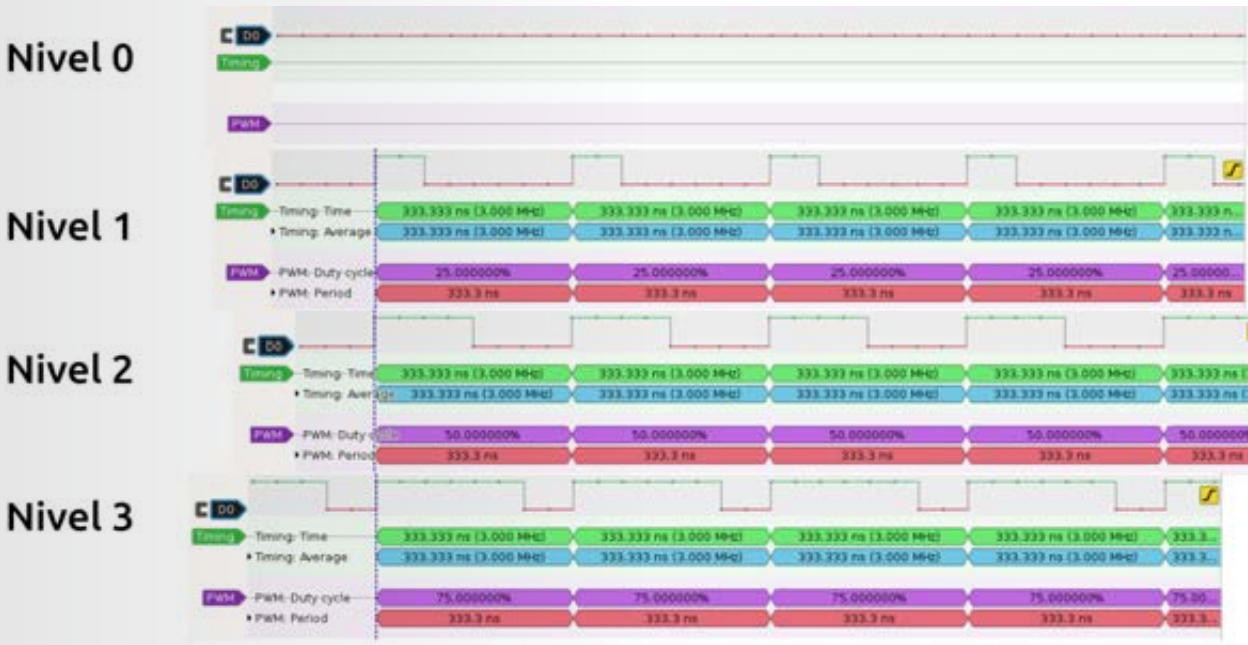


Mediciones

- Medición del nivel 1 (PWM 25%): 100 muestras. Frecuencia muestreo: 24MHz



- Medición de los 4 niveles: Se han tomado por separado y juntado en esta figura



- Resultados:  
Se comprueba que la frecuencia del PWM es de 92.5 Hz y que los niveles medidos efectivamente representan el 6.25%, 25%, 62.5% y 93.75%.

Recurso	Usado	Total	Descripción
FFs	24	1280	Biestables
LUTs	124	1280	Tablas combinacionales
PIOs	12	96	Bloques de entrada/salida
PLBs	38	160	Bloques lógicos
BRAMs	0	16	Bloques de memoria

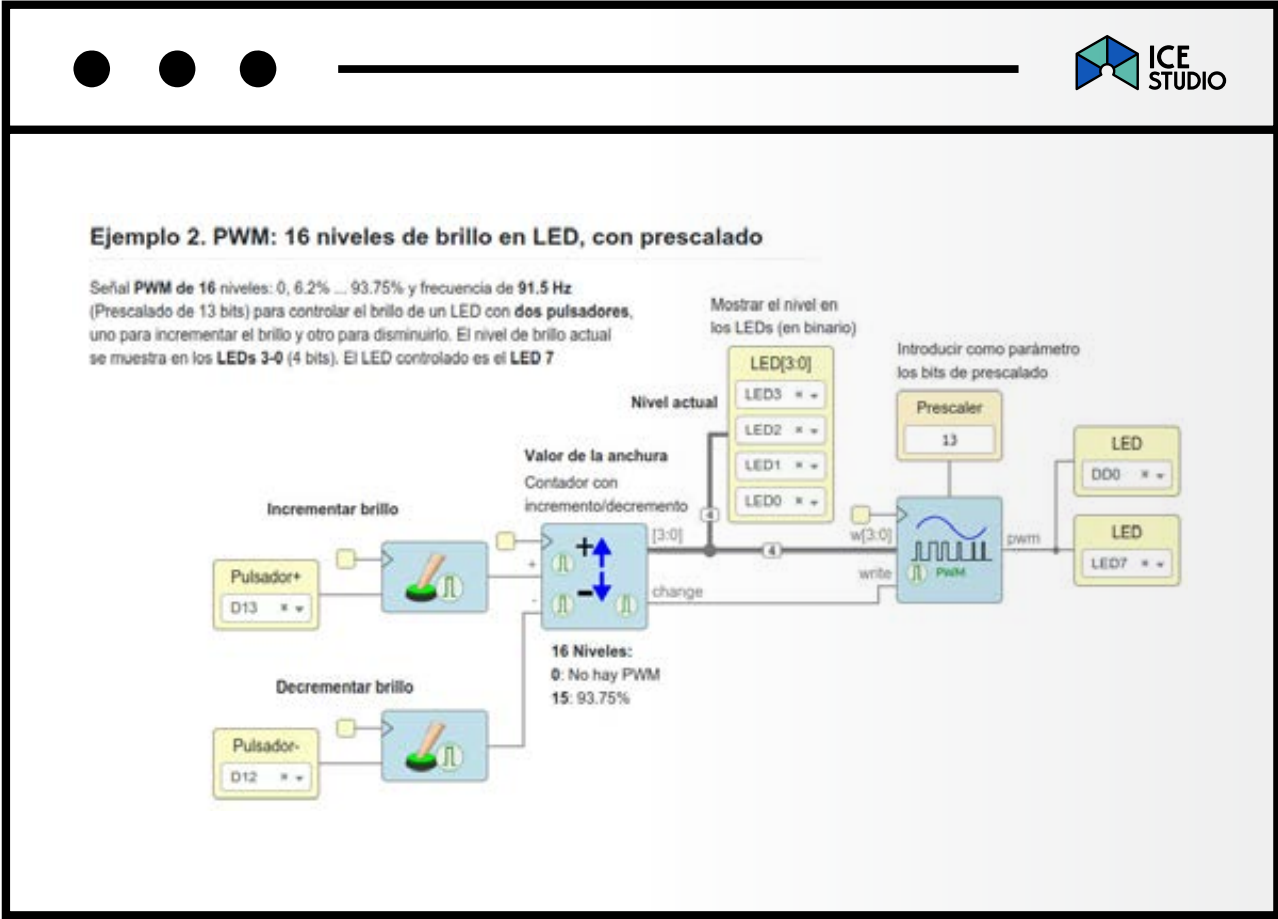


# Ejemplo 2

16 niveles de brillo en LED, con prescalado

Señal PWM de 16 niveles: 0, 6.2% ... 93.75% y frecuencia de 91.5 Hz (Prescalado de 13 bits) para controlar el brillo de un LED con dos pulsadores, uno para incrementar el brillo y otro para disminuirlo. El nivel de brillo actual se muestra en los LEDs 3-0 (4 bits). El LED controlado es el LED 7.

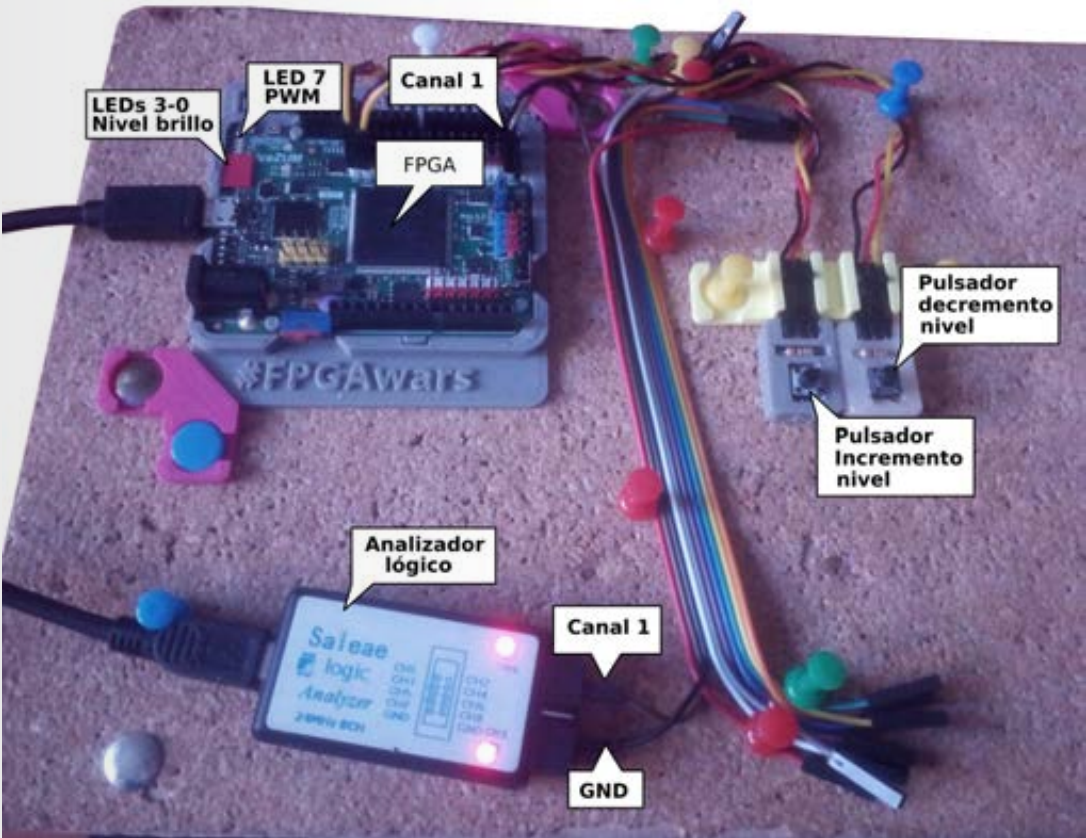
## Circuito



Descargar Fichero  
02-Brillo-LED-pulsadores-4bits.ice

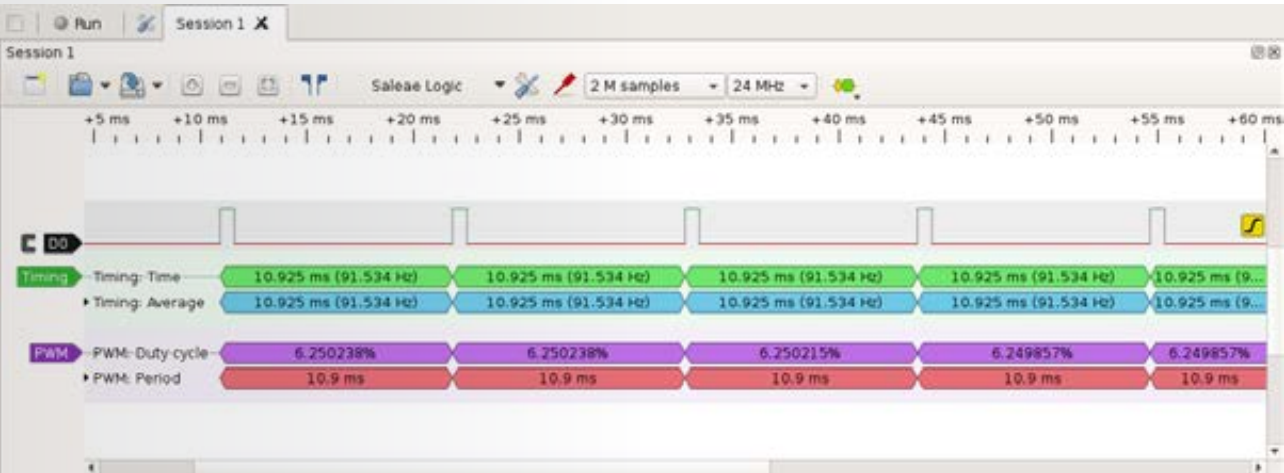
# Escenario

- Elementos: Dos pulsadores externos y el analizador lógico para medir

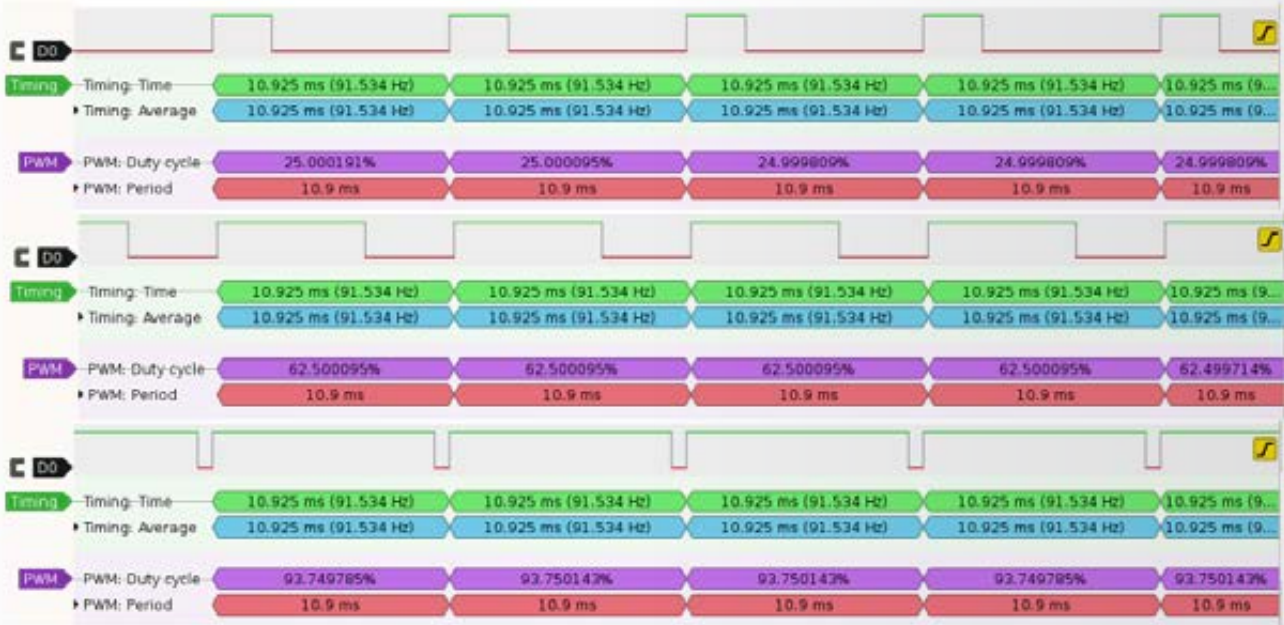


## Mediciones

- Medición nivel 1: (PWM 6.25%): 2M muestras. Frecuencia muestreo: 24MHz



- Medición niveles 4, 10 y 15: Se han tomado por separado y juntado en esta figura



- Resultados: Se comprueba que la frecuencia del PWM es de 92.5 Hz y que los niveles medidos efectivamente representan el 6.25%, 25%, 62.5% y 93.75%

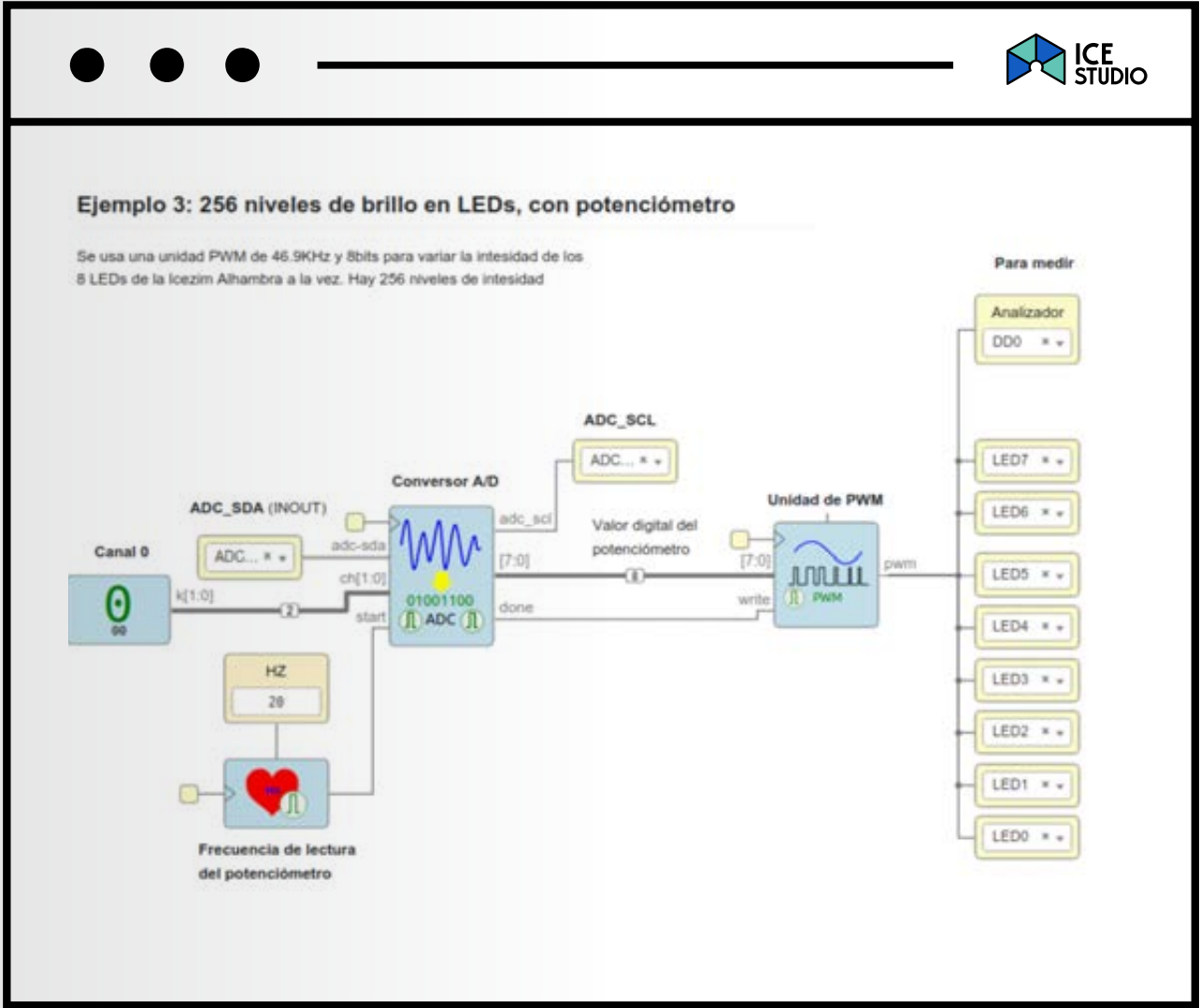
Recurso	Usado	Total	Descripción
FFs	24	1280	Biestables
LUTs	124	1280	Tablas combinacionales
PIOs	12	96	Bloques de entrada/salida
PLBs	38	160	Bloques lógicos
BRAMs	0	16	Bloques de memoria

### Ejemplo 3

Señal PWM de 256 niveles: 0, 0.39%, %99.8 y frecuencia de 46.87 KHz (No hay prescalado) para controlar el brillo de los 8 LEDs con un potenciómetro.

Al girar el potenciómetro a un extremo todos LEDs se apagarán. En el extremo contrario los LEDs brillarán a su máxima intensidad. Variando el potenciómetro ajustamos la intensidad de todos los LEDs. El potenciómetro se lee a una frecuencia de 20Hz.

### Circuito

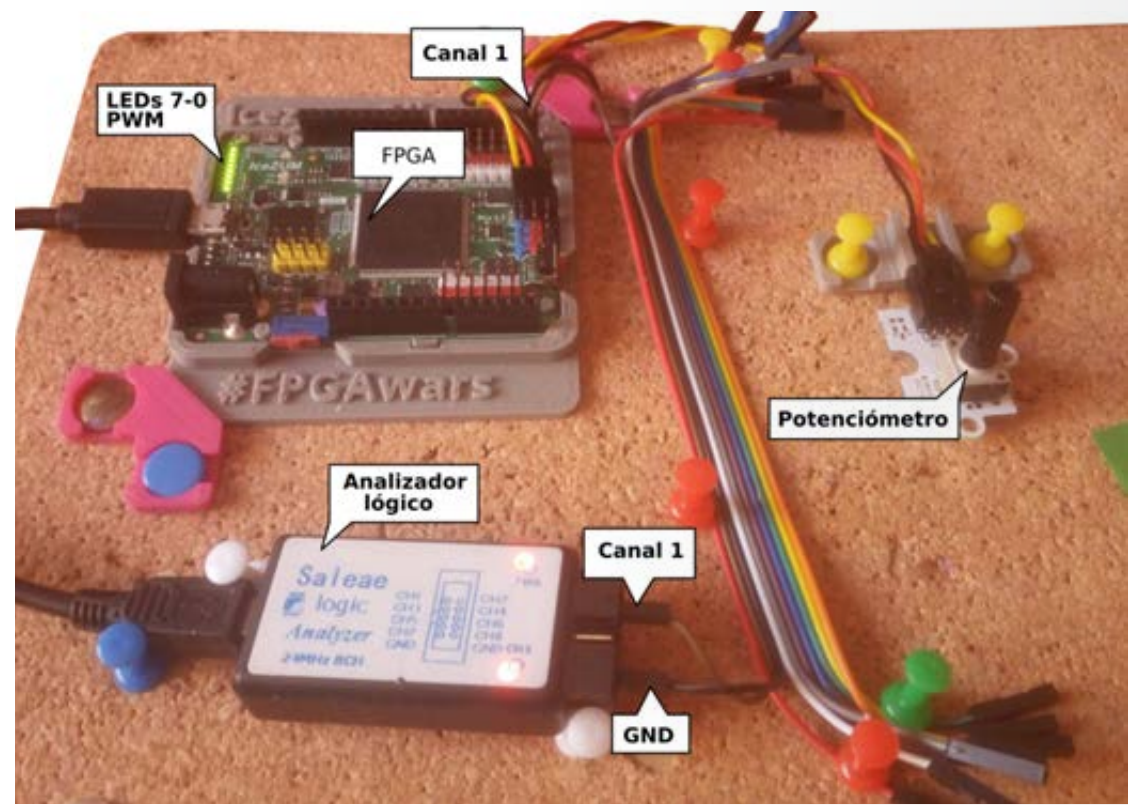


Descargar Fichero  
03-Brillo-LEDs-pot-8bits.ice



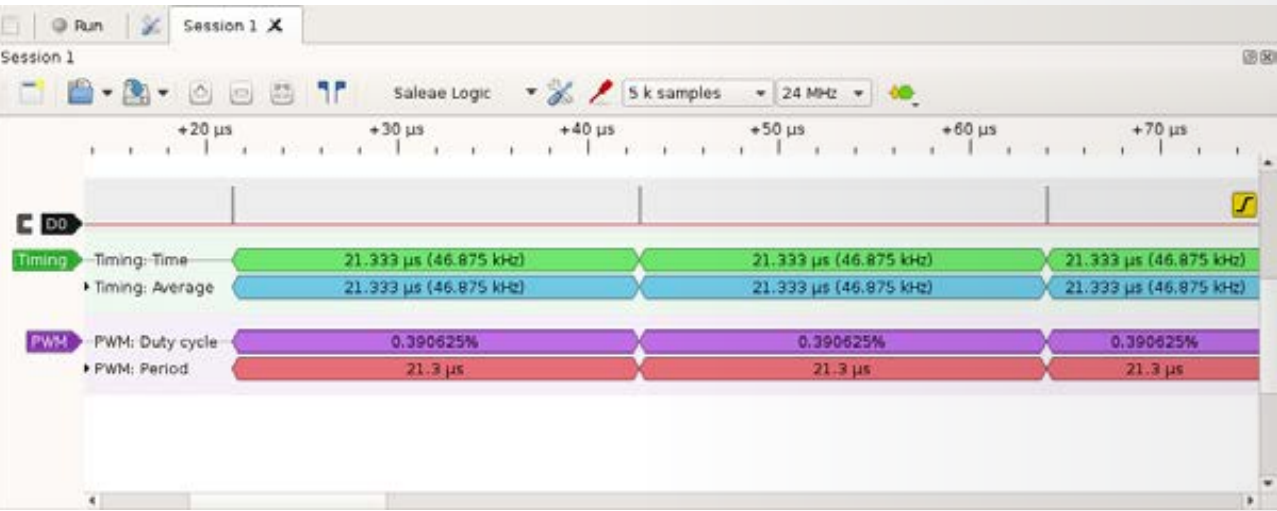
# Escenario

- Elementos: Potenciómetro y analizador lógico para hacer las mediciones

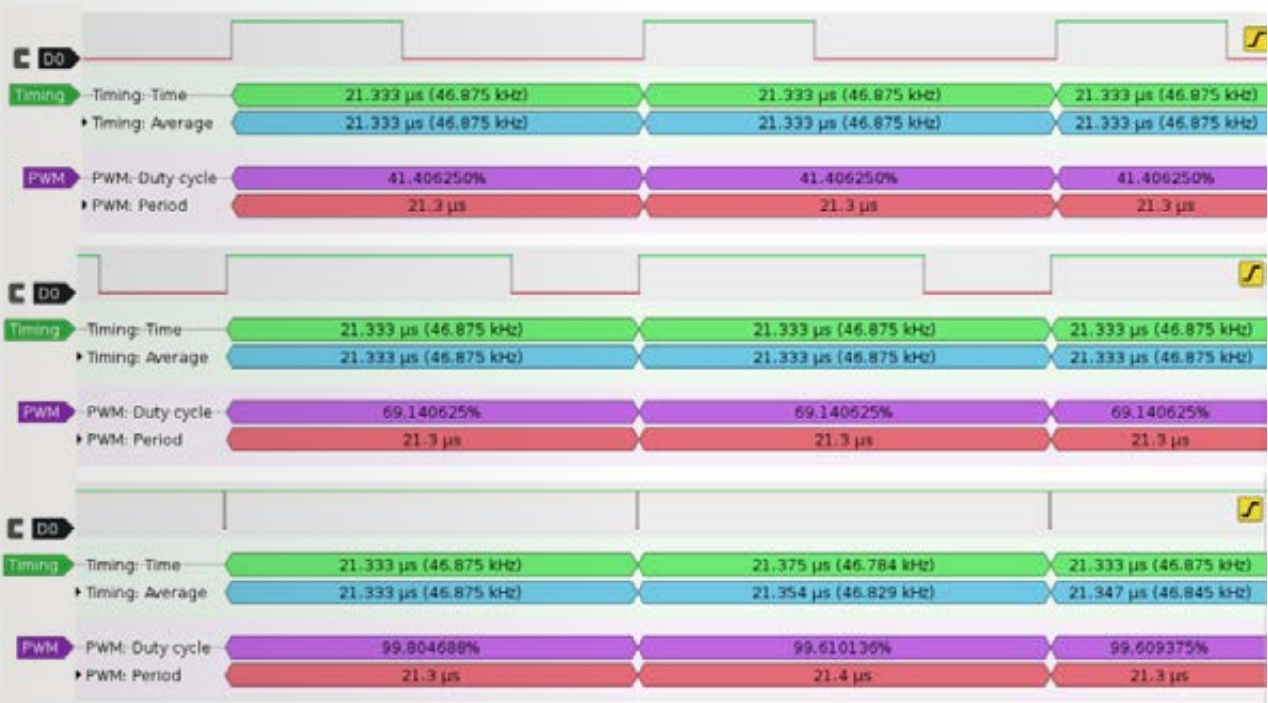


# Mediciones

- Medición nivel 1: (PWM 0.39%): 5K muestras. Frecuencia muestreo: 24MHz



- Medición de tres anchuras: Se han medido por separado y juntado en esta figura



- Resultados: Se comprueba que la frecuencia del PWM es de 46.87 Khz y que los niveles medidos efectivamente van desde el 0.39% hasta el 99.85

Recurso	Usado	Total	Descripción
FFs	116	1280	Biestables
LUTs	309	1280	Tablas combinacionales
PIOs	14	96	Bloques de entrada/salida
PLBs	92	160	Bloques lógicos
BRAMs	0	16	Bloques de memoria



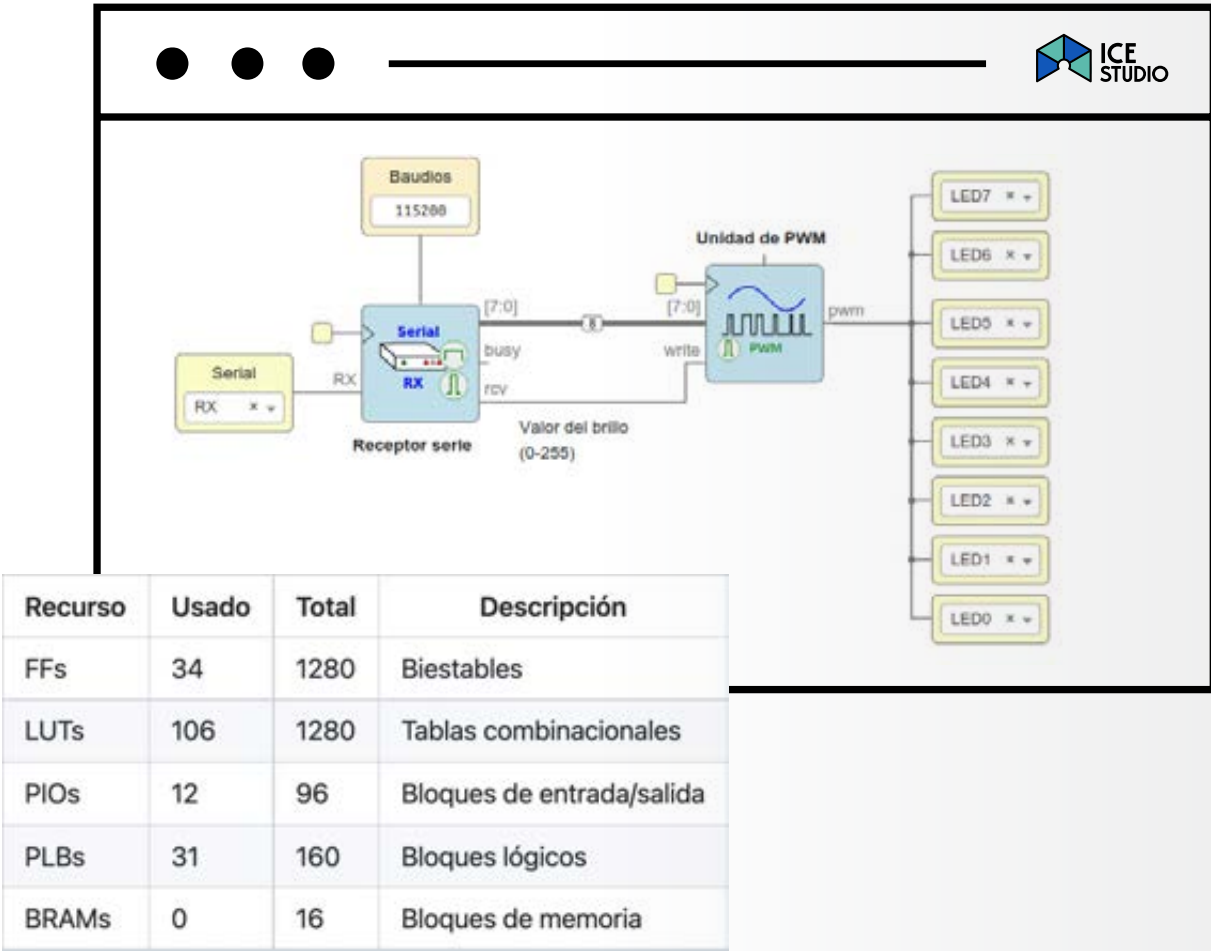
# Ejemplo 4

Servidor hardware de 256 niveles de brillo en LED

Misma señal PWM del ejemplo anterior: Señal PWM de 256 niveles: 0, 0.39%, ..., %99.8 y frecuencia de 46.87 KHz (No hay prescalado) para controlar el brillo de los 8 LEDs. El nivel de brillo (0-255) se envía desde el PC a través del puerto serie.

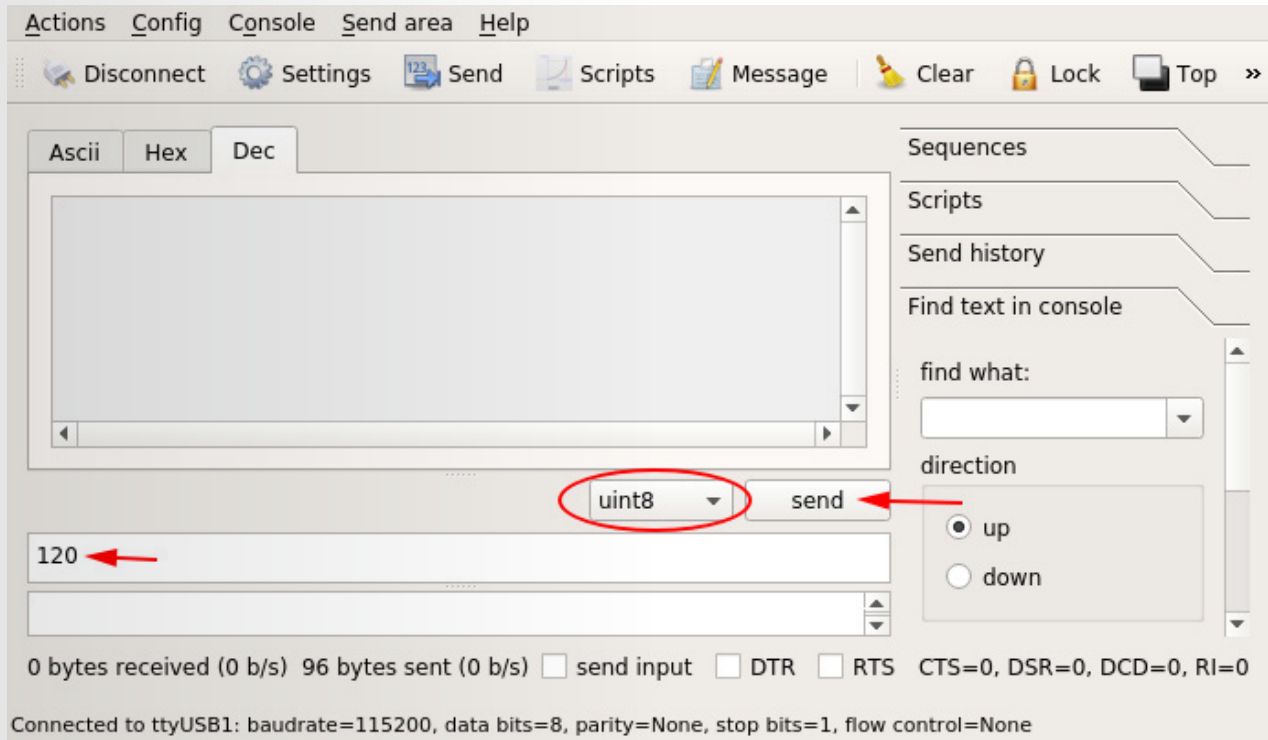
El circuito, por tanto, es un Servidor Hardware que ofrece el servicio de establecimiento del brillo de los LEDs. Lo probaremos desde 3 clientes diferentes: El script Communicator, un programa en python, y un Cuaderno interactivo.

## Circuito

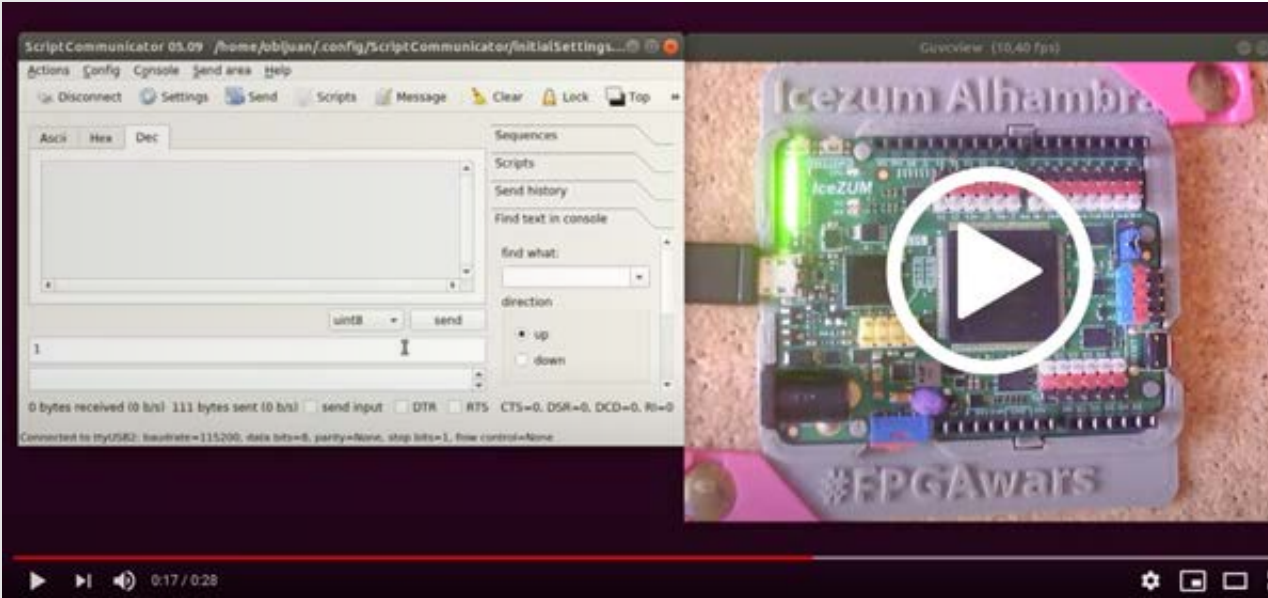


# Cliente 1: Script Communicator

Desde el script Communicator podemos enviar directamente los niveles (0-255) a los que queremos que estén los LEDs. Hay que configurar el envío para que sea de tipo uint8. Se introduce el valor decimal en el recuadro inferior y se aprieta el botón de send




En este vídeo lo vemos en acción

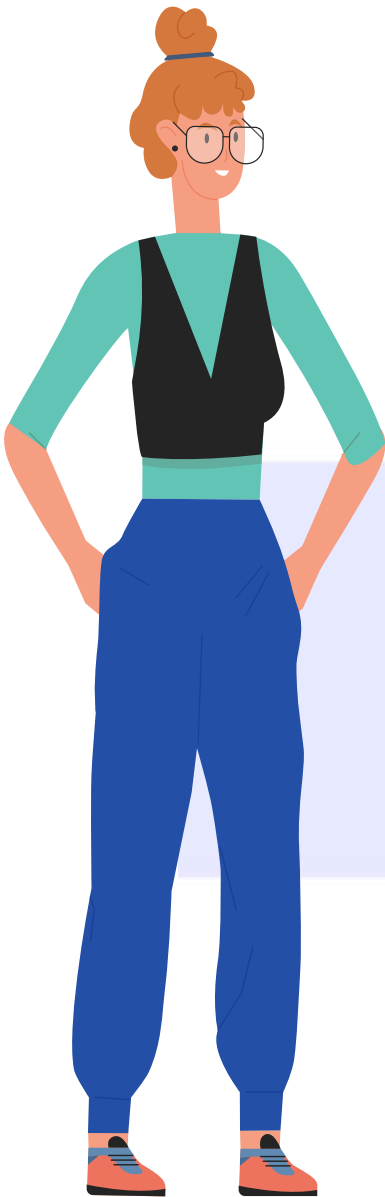
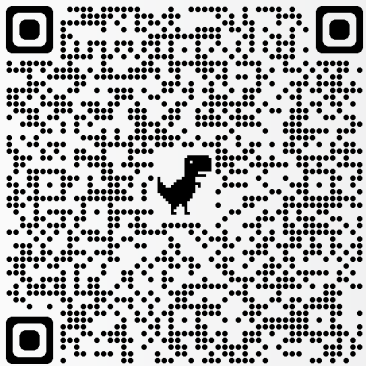


Descargar Fichero  
03-Brillo-LEDs-pot-8bits.ice

# Ciente 2: Python script

Este es un ejemplo de cliente escrito en python, que recorre todas las instensidades, primero desde 0 a 255 y luego al revés, haciendo que los LEDs lleguen a su máximo brillo y luego lo bajen hasta que se apaguen. El patrón se repite varios haciendo.

 Programa ejemplo: pulse-LEDs.py



```
import time
import serial

#-- Poner el nombre del puerto serie aquí
#-- En windows será COMx
SERIAL_PORT = "/dev/ttyUSB2"

#-- Periodo del pulso en segundos (apagado - maximo - apagado)
T = 0.5

#-- Numero de repeticiones
REPETICIONES = 10

#-- Abrir el puerto serie
with serial.Serial(SERIAL_PORT, 115200) as sp:

    #-- Imprimir la información del pueto serie
    print("Puerto serie: {}".format(sp.portstr))

    #-- Relizar dos iteraciones
    for i in range(REPETICIONES):
        #-- Aumentar brillo de 0 a 255
        for brillo in range(255):
            sp.write(bytes([brillo]))
            time.sleep(T/256)

        #-- Disminuir el brillo
        for brillo in reversed(range(255)):
            sp.write(bytes([brillo]))
            time.sleep(T/256)


    #-- Tiempo de espera hasta el siguiente pulso
    time.sleep(1)
```

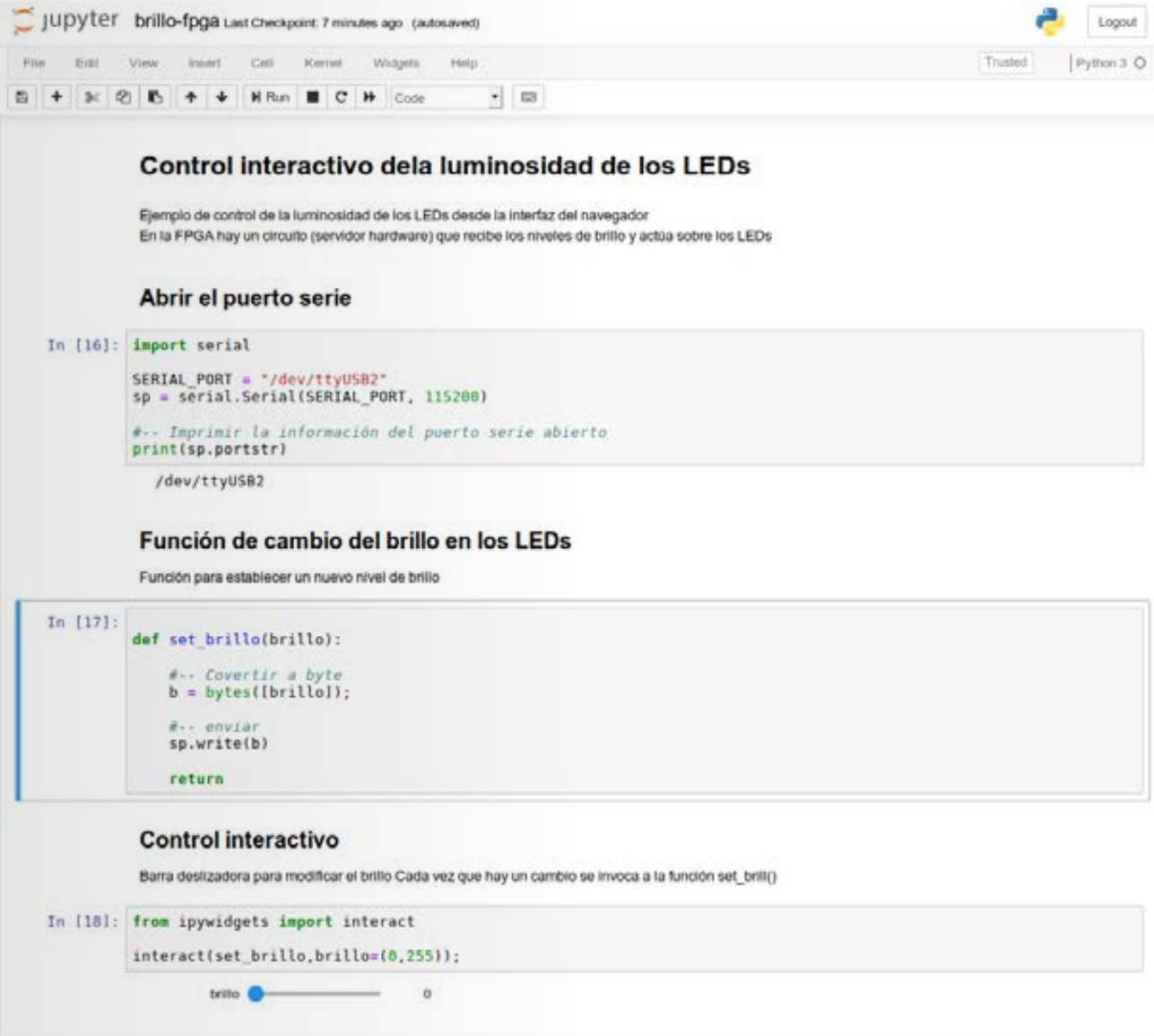


Y en este vídeo lo vemos en acción. El programa se interrumpe pulsando control-c (o bien esperando a que se terminen todos los ciclos)

# Ciente 3: Python notebook (Jupyter)

En el tercer cliente usaremos una barra deslizante en un cuaderno de Jupyter, en Python, para establecer el brillo de los LEDs.

 Descargar fichero brillo-fpga.ipynb





# ¡Gracias!



Comunidad  
libre de  
electrónica  
digital



[fpgawars.github.io](https://fpgawars.github.io)