

Laboratorio 1 de electrónica digital III (sistema de medición de frecuencia)

**JUAN DAVID RIOS RIVERA¹, (1116922203), MARIO ALEJANDRO TABARES OREJUELA²,
(1113788443)**

¹Estudiante Ingeniería Electrónica: (e-mail: juan.rios30@udea.edu.co)

²Estudiante Ingeniería Electrónica: (e-mail: alejndro.tabares@udea.edu.co)

Departamento de Ingeniería Electrónica - Universidad de Antioquia

ABSTRACT En el presente informe se detalla el proceso llevado a cabo para el diseño e implementación de un generador y medidor de frecuencias, implementado en un microcontrolador arduino basado en el microchip ATmega328P, usando el IDE de arduino.

I. INTRODUCCIÓN

Un microcontrolador es un circuito integrado que se puede programar, dicho de otra forma puede ejecutar las órdenes que tenga almacenadas en su memoria. El circuito se encuentra constituido por una unidad central de procesos la cual se encarga de ejecutar las instrucciones y operaciones del programa, a su vez cuenta con memorias de almacenamiento para el guardado de variables, funciones, datos y direccionamientos de los cuales se hacen uso en el programa, por último el integrado consta de periféricos de entradas y salidas que son herramientas propias del él, estos pueden ser utilizados para la solución de alguna actividad. Por ejemplo para la práctica se realizó el montaje de un sistema de medición de frecuencia con base a dos tarjetas arduinos [6], realizando la transmisión y recepción de la señal PWM [7], por medio de los puertos 2 y 9 respectivamente de cada tarjeta. La interfaz gráfica del programa estará lista en todo momento para recibir los comandos (start, stop, reset, duty, freq), cuando por ella se reciba el comando start, se inicializa la generación de la señal cuadrada con los valores de frecuencia y duty por defecto. El programa cuenta con el comando duty, que modifica el ciclo de dureza de la señal actual, lo hace entre un valor del 0-100 porcentaje del tiempo de la señal en alto, por último se tiene el comando freq, este altera el valor de frecuencia por defecto cambiándolo entre un rango de 1- 50KHz que es el escogido por el usuario y así poderlo ver reflejado de manera electrónica en un arreglo de display de 7 segmentos [4] .

II. GENERADOR

Para la implementación del generador de pulsos PWM, se implementó un programa en arduino que recibirá las variables de la terminal, esto con la finalidad de modificar los

parámetros de frecuencia y duty de la señal. El generador se realizó con el uso de la librería Timerone soportada por el microchip ATmega328P. Esta hace uso de interrupciones y modificaciones del registro TCCR1B para cambiar los estados de los valores internos y alterará la frecuencia propia del reloj de arduino.

A. REGISTRO DE ARDUINO Y LIBRERIA TIMERONE

Registro de control B del Timer/Contador1 - TCCR1B: Este registro es una variable de tipo byte la cual podemos leer y escribir para configurar el funcionamiento del microcontrolador a un nivel bajo.

ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
7	6	5	4	3	2	1	0

Este registro [8], está compuesto por 8 bits que desempeñan las siguientes configuraciones.

Bit 7-ICNC1: Entrada de captura1 anuladora de ruido. Cuando el bit 7 es puesto en cero, se deshabilita la activación de la función anuladora de ruido de la entrada de captura, en caso contrario cuatro muestras sucesivas son tomadas en el ICP y todas ellas deben estar en alto/bajo según las especificaciones de activación y de esa forma la frecuencia de prueba del reloj es la del XTAL.

Bit 6-ICESI: Selección de flanco de entrada de captura 1. Mientras el bit 6 es puesto en cero, el contenido del Timer/Contador1 se transfiere al registro de captura de entrada (ICR1) en el flanco de bajada del pin de entrada(ICP). En el caso en el que el bit es puesto en 1, el contenido del timer/contador1 se trasfiere al ICR1 en el flanco de subida

del ICP.

Bit 5: Bit reservado. Este bit es reservado en los AT90S8515 y siempre su lectura es cero.

Bit 4, 3 WGM1:Borrado de Timer/contador1 en la operación de comparación. Cuando el bit de control esta a uno, el timer/Contador1 es reseteado a 0000 en el ciclo de reloj siguiente a la comparación. Si el bit de control WGM1 es puesto en 0 el Timer/Contador1 continua contando y no es alterado por la operación de comparación.

Bits 2, 1, 0: Los bits de CS12, CS11, Cs10 : Selectores de reloj. Los bits 2, 1 y 0 de selección de reloj 1, definen la fuente del prescalar del Timer/Controlador1

CS12	CS11	CS10	Descripción
0	0	0	Stop, el Timer/contador1 esta parado
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Pin externo T1, flaanco descendente
1	1	1	pin externo T1, flanco ascendente

Libreria Timeone: Esta librería es desarrollada originalmente por Jesse Tane y ha sido modificada a lo largo del tiempo con contribuciones de la comunidad, dando soporte a diferentes tipos de tarjetas de microcontroladores como es el caso del microchip del arduino uno. La principal función de esta librería, es la de generar una función cuadrada PWM la cual pueda modificar los valores de frecuencia y duty de la señal. Esto lo realiza modificando los prescalares de tiempo del registro del Timer interno, encontrando unos rangos de resolución por ciclos para diferentes tipos de frecuencias. En la librería [1], encontramos el programa para el soporte de la tarjeta del arduino, desde la linea de código 172 hasta la linea 296.

B. COMANDOS DE EJECUCIÓN

El programa del generador de señales consiste en un loop interno con dos funciones principales, la primera se encarga de crear la terminal por la cual se reciben los comandos verificados del programa y la segunda se encarga de mostrar activa o desactiva la señal PWM. Principalmente en la ejecución del programa del arduino en la parte del setup se

ejecuta la función de la librería Timeone:

Timer1.setPwmDuty(PWM , dutyescalado);

Esta recibe como parámetros de entrada el pin por el cual va a generar la salida del PWM y el valor del duty con el que arrancara la señal.

* Si el comando es START: Activa una bandera de nombre active poniéndola en alto para indicar que esta activo el sistema.

* Si el comando es STOP: Lleva el valor de 0 a la bandera de active que desactiva el sistema y hace uso de la función de la librería Timeone.

Timer1.disablePwm(PWM)

Esta deshabilita el pin digital de la salida de la señal del arduino

* Si el comando es freq seguido de un espacio y el valor: El programa entra en un condicional que almacena el valor ingresado y hace uso del comando de la librería Timeone Timer1.initialize(freqenHz)

Esta función lo que hace es cambiar el valor de frecuencia con el que se inicializa la librería, encontrando la resolución de ciclos que mejor se adapte para ser mostrada por el pin digital.

* Si el comando es duty seguido de un espacio y el valor: El programa entra al condicional para almacenar el nuevo valor de duty y modifica su valor al llamar a la función de Timerone

Timer1.setPwmDuty(PWM, DutyNuevo)

Esta función modifica el valor del duty de la señal, recibiendo como parámetros el pin de salida de la señal y el valor modificado del duty

* Si el comando es reset: El programa vuelve a inicializar los valores por defecto que tenia el sistema.

La segunda función que determina si sistema esta activo o desactivado hace uso de la función de la librería Timerone.

Timer1.pwm(PWM, duty)

Esta función recibe como parámetro la salida de pin digital y el duty de la señal cuadrada.

III. MEDIDOR DE FRECUENCIA

Para la implementación del medidor de frecuencia, se plantea el uso de interrupciones con el fin de contar los pulsos que se presentan durante un segundo, ya sea con el flanco de bajada o subida, al contar los pulsos por un segundo tenemos la frecuencia en Hz. Para ello es necesario comprender como funcionan las interrupciones en el arduino. Una vez obtenida la frecuencia se procede a ser mostrada en cinco display 7 segmentos, siendo estos multiplexados para que sean observados como si todos estuvieran activados al mismo tiempo. En el loop del arduino se crea un condicional que evalúa si el tiempo de refresco se cumple, si es el caso, se evalúa la medida de frecuencia, de lo contrario continúa multiplexando

la frecuencia ya guardada, además se evalúa cual es la nueva frecuencia para saber si la tasa debe ser diferente a la que hay presente o la misma.

A. INTERRUPCIONES DE ARDUINO

Las interrupciones en arduino nos permiten darle prioridad a una acción dado que se presente un evento en uno de los pines digitales en arduino las interrupciones pueden ser tomadas en los pines 2 y 3 en la placa arduino Uno sobre la cual esta basada este trabajo [4]. En este caso se usa el pin 2, al cual después de activarle las interrupciones se le asigna una función la cual se ejecuta una vez la interrupción se active, siendo lo mas simple posible es solo un contador que se incrementa. Esto debido a que para asignar la interrupción se sigue la instrucción

```
attachInterrupt(digitalPinToInterruption(pin), ISR, modo  
(1)
```

) donde el pin es 2, ISR es la función que se ejecuta cada vez que hay una interrupción, en este caso la función se llama countPulse(), no recibe parámetros ni retorna nada, solo dentro se encuentra la variable volátil contadora que se incrementa. Es una variable volátil dado que este tipo de variable pueden ser consultadas antes de ser usadas, ya que existe la posibilidad de haya sido modificada en otra instancia del programa, finalmente el modo de la interrupción es cuando el flanco esta en subida, es decir RISING, por tanto el numero de pulsos en subida que se detecten en un segundo nos dará la frecuencia.

B. TASA DE REFRESCO

En búsqueda de cumplir las especificaciones de diseño requeridas, donde es necesario una tasa de refresco de la frecuencia leída de 1Hz para frecuencias menores a 4Hz y de 4Hz para frecuencias mayores a este mismo valor de frecuencia leída. Esto traducido a tiempo sería leer la frecuencia cada segundo si es menor o igual a 4Hz y cada 250 milisegundos si es mayor a 4Hz, esto de vera reflejado en como fluctúa el valor visto en los display 7 segmentos, cambiando mas rápidamente en frecuencias mas altas.

IV. MULTIPLEXACIÓN EN DISPLAY 7 SEGMENTOS

En esta sección se explica el proceso llevado a cabo para la multiplexación, la cual se hace secuencial, con intervalos de 1ms, entre los displays, teniendo de esta forma una visualización correcta de los mismos, es decir, se imprimen las unidades, luego las decenas, centenas, miles y diezmiles a intervalos de 1ms. Cada uno estaría siendo multiplexado a 1KHz, pero al ser 5 display esta frecuencia se divide así que cada display está a una velocidad de 200Hz, puede parecer excesiva pero mas adelante se detalla el porqué de su reducción. Para la correcta escritura de los números se tienen definidas las variables tipo arreglo de enteros, con formato led a, led b, led c, led d, led e, led f, led g donde cada led del 7 segmentos se enciende con un 0, por ser ánodo común. Del mismo modo para que se encienda uno solo de

los displays con el numero correcto se tiene un arreglo de entero con 5 posiciones, puesto que son 5 leds, con formato display 1, display 2, display 3, display 4, display 5, donde se enciende los primeros 2 con un 1 por tener transistores NPN, los siguientes 2 con 0 por ser PNP y el final con 1, por ser igualmente NPN.

La multiplexación se realiza en todo momento, es por ello que mientras se ejecuta la espera de un segundo para refrescar la frecuencia, la anterior se continúa multiplexando. Por su parte para saber que número escribir en cada display se usa una función que nos brinda en variables separadas cada dígito del número a dibujar para posteriormente seleccionar el arreglo correspondiente a dicho numero, dibujarlo y activar el display. En los arreglos de los números se incluye el guión bajo, siendo solo el segmento "d" de los displays y el guión alto a su vez el segmento "a", para encenderlos cuando la frecuencia sea menor a 1Hz o mayor a 50KHz respectivamente.

El correcto flujo entonces de la multiplexación es:

- 1) Se lee verifica que la tasa de refresco se cumpla
- 2) Se lee la nueva frecuencia, con la función espera()
- 3) Se evalú que valor tiene la nueva frecuencia y se actualiza la tasa de refresco para la próxima iteración
- 4) Se inicia la multiplexación
- 5) Se toma el valor de la frecuencia y se separan sus dígitos
- 6) Se evalúa, uno a uno, si cada dígito es apto para escribirse teniendo en cuenta las reglas, que no deben existir ceros a la izquierda, ni frecuencias mayores a 50000, ni menores a 1.
- 7) Se dibuja cada digito despues de su evaluación condicionada, seleccionando el arreglo correspondiente a los leds del display que deben estar activos, y el arreglo correspondiente a encender dicho display y apagar los demás
- 8) Se deja encendido 1ms y se apaga el display
- 9) Se continua con los demás displays, en orden desde las unidades hasta los diezmiles
- 10) Se vuelve al loop infinito donde se sigue evaluando si la frecuencia ya se debe leer, de lo contrario solo se multiplexa.

V. ANÁLISIS DE RESULTADOS

En cuanto al medidor de frecuencia se observa que para frecuencias menores a 1, se dibuja en los displays el segmento d como se observa en la figura ??, siendo este uno de los requisitos de las práctica.

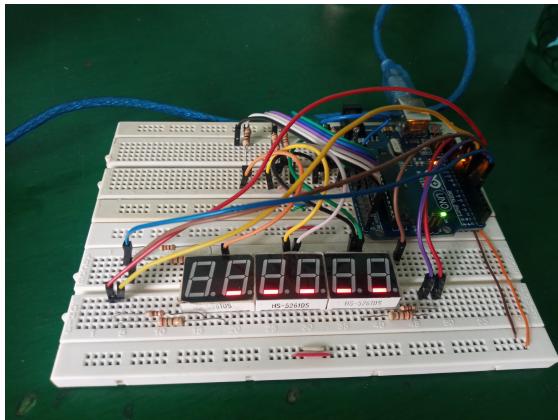


FIGURE 1: Frecuencia menor a 1Hz

Por su parte para frecuencias mayores a 50KHz se dibujan los segmentos a de los displays 7 segmentos como se observa en la figura 2.

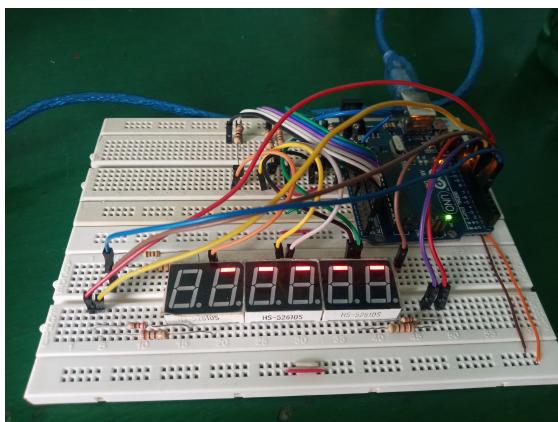


FIGURE 2: Frecuencia mayor a 50KHz

La mayor frecuencia que se recibe, ya que se encuentra desfasada a mayor sea la frecuencia, por la exactitud del medidor es cuando se envía una frecuencia desde el generador de 47.6KHz, donde en los displays se muestra una frecuencia leída de 49985Hz, como se observa en la figura 3.

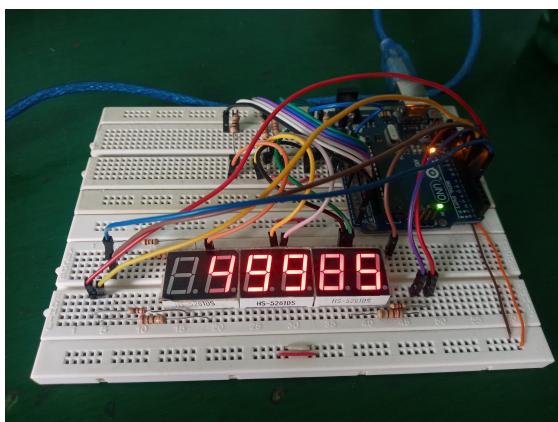


FIGURE 3: Frecuencia máxima leída, enviada 47.6KHz

VI. CONCLUSIONES

- Se destaca de la práctica la importancia del funcionamiento de bajo nivel, esto debido a que sin la intervención de los registros propios del reloj del microcontrolador, la generación de la señal por medio de retardos y estados de alto/bajo en un ciclo infinito se vuelven obsoletos, ya que cuando se varía la frecuencia los valores de aumento o disminución de esta, son múltiplos escalares de la frecuencia interna programada en el integrado.
- La resolución del medidor de frecuencias se ve afectada por los delay que se encuentra presente en la multiplexación durante la etapa de lectura de los pulsos de subida, siendo esta mayormente afectada a mayores frecuencias. Esto debido a que mientras se leen los pulsos de subida se aumenta el tiempo de lectura en algunos milisegundos extras, lo cual hace que en esos milisegundos, a partir de frecuencias de los KHz se detecten pulsos extras que aumentan la medida de la frecuencia.
- El medidor de frecuencia puede mejorarse si no se introducen estos delays que se plantean en la multiplexación cambiando la lógica del programa, e introduciendo mejoras, como el conteo de espacios de tiempo con las funciones `micros()` o `millis()` que no detienen por completo la ejecución del código alargando el tiempo de ejecución mismo.
- La multiplexación a partir de 3ms de espera entre uno y otro display se aprecia de manera excelente, la disminución hasta 1ms se da principalmente para mejorar la resolución de nuestro medidor de frecuencias.

REFERENCES

- [1] PaulStoffregen. (2019). TimerOne/TimerOne.h at master · PaulStoffregen/TimerOne. Recuperado de <https://github.com/PaulStoffregen/TimerOne/blob/master/TimerOne.h>
- [2] PJRC. TimerOne TimerThree Arduino Libraries. Recuperado de https://www.pjrc.com/teensy/td_libs_TimerOne.html
- [3] ElectroDaddy. 29 - Los Registro de Arduino. Recuperado de <https://www.electrodaddy.com/los-registros-de-arduino/>
- [4] C, S. (2019). Interrupciones Arduino - [Que son y como usarlas]. Recuperado de <https://controlautomaticoedificacion.com/arduino/interrupciones-arduino/>
- [5] C, S. (2018). Multiplexar Display 7 Segmentos con Arduino. Recuperado de <https://controlautomaticoedificacion.com/arduino/multiplexardisplay7-segmentos/>
- [6] ARDUINO UNO R3(2023).ATmega328P and the ATmega 16U2 Processor. <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>
- [7] Electronica Unicrom. Recuperado de. <https://unicrom.com/pwm-modulacion-por-ancho-de-pulso/>
- [8] Timer arduino ATmega328P. Recuperado de <https://controlautomaticoedificacion.com/arduino/timer-arduino/>

...