Práctica 9

Generador de Frecuencia mediante los Temporizadores del uC ESP32

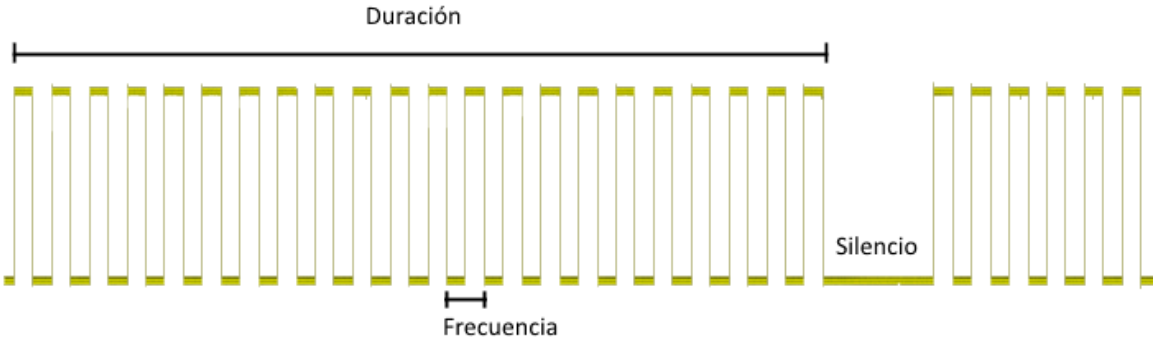
Objetivo: Mediante esta práctica el alumno aprenderá la programación y uso avanzado del Temporizador 0 y 1 del microcontrolador ESP32.

Material:
- Tarjeta de desarrollo ESP32.
- Componentes Electrónicos.

Teoría:
- Teoría Básica de Música.
- Programación del Timer0-1 del microcontrolador. (Diagrama, Funcionamiento, Registros de configuración y operación)

Desarrollo:

El propósito de la practica es la reproducción de una canción en base a tonos, mediante el generador de frecuencia. En la siguiente figura se puede visualizar lo que llamaremos una *nota*:



Donde cada nota estará dada por una frecuencia y una duración, como se muestra en la siguiente estructura:

```
struct note{
      uint16_t freq;
      uint16_t delay;
};
```

Y una canción va a estar dada por un arreglo de notas. Es importante tomar en cuenta que es necesario insertar un tiempo de **silencio** entre cada nota con una duración de **10 ms**, para poder distinguir cuando inicia y termina una.

Para poder llevar a cabo lo anterior es necesario hacer uso de los Timer0 y Timer1 conjuntamente, y para esto se requiere realizar los ajustes necesarios para que el Timer1 se encargue de generar la frecuencia, donde la salida del generador se verá reflejado en el GPIO de su elección, y el encargado de llevar el conteo del tiempo con base de 1 ms será el Timer0.

Las frecuencias de las notas usadas como ejemplo son las siguientes:

```
#define c 261
#define d 294
#define e 329
#define f 349
#define g 391
#define gS 415
#define a 440
#define aS 455
#define b 466
```

```
#define cH 523
#define cSH 554
#define dH 587
#define dSH 622
#define eH 659
#define fH 698
#define fSH 740
#define gH 784
#define gSH 830
#define aH 880
```

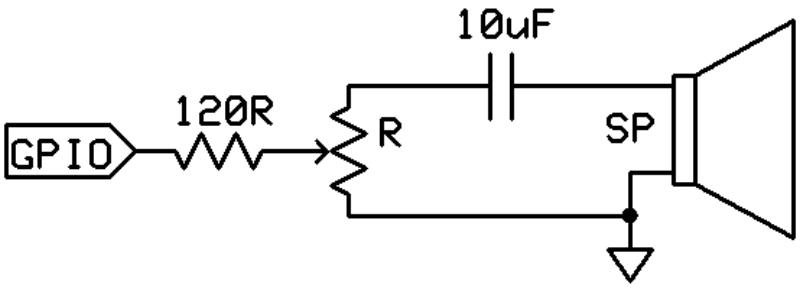
Las cuales las puedes encontrar en el siguiente enlace:
http://www.intmath.com/trigonometric-graphs/music.php

Y el listado del arreglo de notas de ejemplo lo pueden encontrar en el repositorio de la práctica.

A continuación, se presenta el listado de myTimer.c, con las descripciones de las funciones y macro que se solicitan para esta práctica.

```
void timer0Init ( void ){
  /* Configura el Timer0 para generar interrupciones cada
     milisegundo */
}

static void IRAM_ATTR timer0Isr(void *ptr){
/* Código para actualizar bandera de segundos */

/* Agregar las instrucciones necesarias para reproducir
    la siguiente nota en el arreglo dependiendo de la duración,
    e insertar los silencios entre cada nota. */
}

void IRAM_ATTR timer1FreqGen(int16_t freq){
  /* Si "freq" es mayor que 0 entonces, inicializa y habilita el
    Timer1 con la alarma calculada en base a "freq".
    De lo contrario se requiere deshabilitar, generando de
    esta forma el silencio (0 lógico). */
}

void timer1Play(const struct note song[],uint16_t len){
  /* Función que establece las condiciones necesarias para que
    el generador recorra el arreglo de notas. */
}
```

Una vez codificadas las funciones anteriores, ahora es necesario alambrar el siguiente diagrama para poder conectar la tarjeta de desarrollo a una bocina (4Ω-8Ω) y lograr escuchar las frecuencias generadas.



Agregar por lo menos una canción mas en memoria de programa, e implementar el cambio de canción.

Comentarios y Conclusiones.

Bibliografía.