Arquitectura de Computadoras 2019

Trabajo de Laboratorio 2:

OPCIÓN 1: 1-bit audio con RPi3 mediante el uso de GPIO y Timer

Objetivo

Utilizando los conceptos aprendidos de manejo de puertos I/O e interrupciones, aplicados a la programación de procesadores ARMv8, se propone utilizar una plataforma real (Raspberry Pi 3/3+) y hacer uso de bloques periféricos (GPIO) y ARM Timer para generar un respuesta audible controlada con un pin, con el cual se controla el cono del parlante en forma directa mediante una secuencia de '1's y '0's (técnica conocida como 1-bit audio).

Condiciones

- Realizar el trabajo práctico en grupos de **máximo 3 personas**.
- Enviar el trabajo resuelto a agustin.laprovitta@gmail.com. La fecha límite de entrega es el **miércoles 13 de Noviembre** (inclusive). Los trabajos entregados después de esa fecha se consideran desaprobados.
- Defender en un coloquio el trabajo presentado. Deben presentarse todos los integrantes del grupo y responder preguntas acerca del trabajo realizado. El coloquio es el **viernes 15 de Noviembre**, en el horario del práctico. En caso de no poder asistir al coloquio en esa fecha, coordinar **previamente** con los docentes.
- El número de grupos que realicen el laboratorio está limitado por el hardware disponible para préstamo (Rpi3 + Amplificador de audio). Cada grupo puede utilizar su propia placa Rpi3 en caso de poseer una.

Formato de entrega

Deben entregar un tarball con el nombre:

`TPRPi3_Apellido1_Apellido2_Apellido3_Apellido4.tar.gz`, respetando mayúsculas y minúsculas. El tarball debe contener dos archivos con la siguiente denominación: `ejercicio1.s`, `ejercicio2.s`, `etc`. Estos archivos deben seguir el estilo de código del programa del ejemplo y contener comentarios que ayuden a comprender la manera en que solucionaron el problema. En el inicio de cada archivo describir en pocas líneas el efecto sonoro que genera el código.

Calificación

El ejercicio 1 es obligatorio. Su resolución y presentación en coloquio debe estar aprobados para obtener la regularidad de la materia. Quienes además resuelvan el ejercicio 2 obtendrán 1 punto extra para el segundo parcial y si resuelven los 3 ejercicios obtendrán 2 puntos extra para el segundo parcial.

Aunque no se califica: estilo de código, simpleza, elegancia, comentarios adecuados y velocidad; si el código está muy por fuera de los parámetros aceptables, se podrá desaprobar el trabajo aunque sea funcionalmente correcto.

Introducción

Una onda de sonido se produce cuando un parlante es excitado por una señal eléctrica alternante en el tiempo, es decir que el flujo de corriente eléctrica en su bobinado cambia de sentido a lo largo del tiempo (Ver Fig. 1). Cuando la corriente eléctrica fluye en el sentido +/- (semiciclo positivo de la onda senoidal de la Fig. 2), el cono del parlante avanza y genera una onda de presión de aire. En cambio si la corriente fluye con dirección -/+ (semiciclo negativo de la onda senoidal de la Fig. 2), el cono retrocede y genera una de depresion.

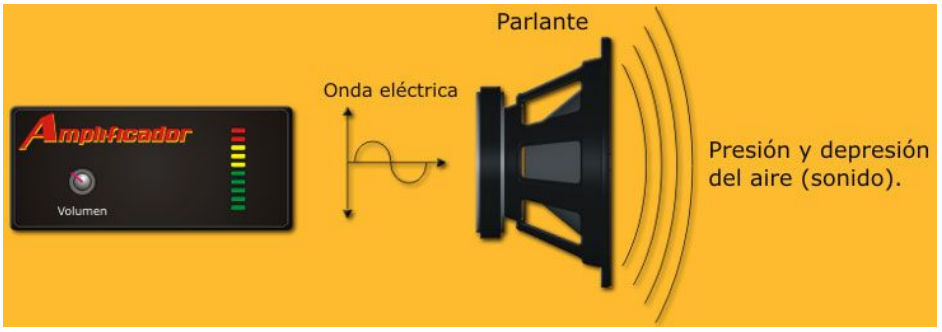

Fig 1. Esquema de excitación de un parlante

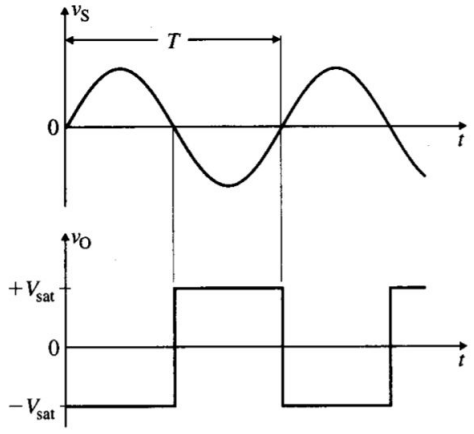

Fig 2. Ondas periódicas. (sup) Onda senoidal, (inf) Onda cuadrada

La distancia o recorrido del cono del parlante es proporcional a la amplitud de la señal que excita el parlante, determinando el volumen del sonido generado. Mientras que la cantidad de veces que el parlante avanza y retrocede en un periodo de tiempo dado determina el tono del sonido. Normalmente este periodo de tiempo es 1 seg., y esta velocidad de cambio (o frecuencia) se mide entonces en Hertz [Hz]. Si en el ejemplo de la figura T = 1 seg, se dice que esa onda tiene una frecuencia de 1 Hz. En cambio, si T fuese de 0.5 seg, la frecuencia sería de 2 Hz. Es decir que:

$$f\,[Hz] = 1/T$$

Este laboratorio está basado en la técnica de generación de sonido conocida como “1-bit audio” y hace referencia a la limitación de solo poder generar una onda cuadrada de amplitud fija para excitar el parlante (ver onda inferior de Fig 2). De esta forma, si la señal de control del parlante es:

“1” → parlante avanza

“0” → parlante retrocede

En esta técnica sólo es posible alterar el ritmo del cambio de la señal, pero no su amplitud, es decir, solo podemos controlar su frecuencia.

La variedad de tonos que nuestro oído es capaz de percibir es muy elevada, estando acotada tan sólo por los límites de sensibilidad de nuestro sistema auditivo, normalmente desde los 20Hz hasta los 20kHz. La gama usual de frecuencias de los sonidos musicales es considerablemente más pequeña que la gama audible, siendo el tono más alto de un piano el de frecuencia 13.186kHz, valor que podemos considerar como límite superior de los tonos fundamentales.

La nota de referencia del sistema musical occidental es actualmente el “LA” a 440Hz (porque la Organización Internacional de Estandarización así lo fijó en 1955). A partir de esta nota de referencia se calculan las frecuencias que deben tener el resto de notas para que todos los instrumentos suenen correctamente afinados.

Las frecuencias de las notas musicales en la octava correspondiente al LA a 440Hz son:

Nota en español	DO	RE	MI	FA	SOL	LA	SI
Nota en inglés	C	D	E	F	G	A	B
Frecuencia [Hz]	261.63	293.66	329.63	349.23	392.00	440.00	493.88

Hay mucho hecho y escrito respecto a cómo generar **1-bit audio**. Varias microcomputadoras de principios de los 80’s traían exactamente el mismo soporte que este laboratorio, un ‘1’ en un puerto levantaba el altavoz, un ‘0’ lo bajaba. Los ejemplos clásicos son Apple][(1977) y ZX Spectrum (1982). Más adelante se incorporaron PSG (programmable sound generators), circuitos integrados específicos que generaban el audio, ejemplos típicos son el SID de la Commodore 64 y el AY-3-8910 de la MSX. Finalmente con la introducción de Paula de Commodore Amiga, se inició la posibilidad de tocar samples de 8 o más bits de profundidad en varios canales. Actualmente esta es la forma de generar sonido en todas las computadoras, un procesador especial a través de acceso directo a memoria (DMA) reproduce samples en la salida de audio.

En este artículo, Kenneth McAlpine explora parte de la historia y las técnicas utilizadas por las computadoras con audio de 1-bit de profundidad.

- Kenneth B. McAlpine, *The Sound of 1-bit: Technical Constraint and Musical Creativity on the 48k Sinclair ZX Spectrum*, The Italian Journal of Game Studies, 6/2017.

Las restricciones respecto a la capacidad de hacer sonido jamás fueron un impedimento. La Altair 8800, una de las primeras microcomputadoras accesible al bolsillo de la clase media, pudo entonar Daisy Bell como lo hizo la IBM 7094 en 1961. Mucho antes, en 1951, un tal Alan Turing ayudó un tal Christopher Strachey a registrar la primera melodía interpretada por una computadora.

- Steve Dompier, *Music of a sort*, Altair 8800, DrDobbs, 1976.

La música 1-bit sigue viva, como en el proyecto 1-bit Symphony de Tristan Perish, o en las demos extremas que combinan muchas Apple][o muchas ZX Spectrum para hacer sinfonías.

En la demoscene local se destacan los Pungas de Villa Martelli (PVM), un grupo que crea arte a partir de tecnologías obsoletas. Aunque no producen piezas de 1-bit audio, los PVM recrean a través de PSG (programmable sound generators) de Commodore 64 y Nintendo Gameboy todo un repertorio vernáculo, como el Cancionero Argentino Vol 1 y 2.

Tal vez la mejor referencia sea el libro recientemente editado

- Kenneth B. McAlpine, "Bits and Pieces, A History of Chiptunes", OUP, 2018

DESARROLLO

Ejercicio 1 (para regularizar)

Escribir un programa en lenguaje C, sobre el código de ejemplo dado, que genere una secuencia infinita de las notas DO - RE - MI - FA - SOL - LA - SI en la octava dada en la introducción. Cada nota debe reproducirse durante aproximadamente 5 segundos.

Cabe señalar que todo el control de tiempos del programa deben ser generados mediante la utilización del módulo ARM Timer por interrupción. **No están permitidos delay generados por código**.

TIP: Para medir la frecuencia de la nota que se está generando utilizar el Afinador Cromático Gismart (en el modo "afinador cromático").

Ejercicio 2 (+1 punto extra)

Escribir un programa en lenguaje C, sobre el código de ejemplo dado, que genere un efecto sonoro "llamativo" o melodía de duración no menor a 30 segundos. Se valorará especialmente la relación del efecto sonoro logrado vs. el tamaño del código generado.

Ejercicio 3 (+1 punto extra)

Escribir un programa en lenguaje C, que genere al menos 3 melodías distintas de duración no menor a 20 segundos c/u. Mediante la utilización de un pulsador conectado a un GIPO se debe poder controlar qué melodía se reproduce. De esta forma, cada vez que se presione el pulsador, se debe cambiar de melodía y pasar a la siguiente en forma cíclica. Si cualquier melodía llega a su fin antes de presionar el pulsador, debe comenzar nuevamente. Se valorará especialmente la relación del efecto sonoro logrado vs. el tamaño del código generado.

Antes de comenzar

1) Cada grupo deberá contar con una computadora, un lector de SD o microSD y un cargador de celular tipo micro USB.

2) Verificar que la SD esté en formato FAT32 y con los archivos de booteo. En caso de no ser así, formatearla, descargar el template del moodle y copiar el contenido de la carpeta boot en el directorio raíz de la tarjeta.

3) Instalar el **GNU Aarch64 toolchain**. Quienes usan Ubuntu, lo pueden bajar del gestor de paquetes con el comando:

```
$ sudo apt install gcc-arm-none-eabi
```

4) Descargar el template del moodle.

Para escribir el código en C

- La aplicación debe ser escrita íntegramente en el archivo `main.c`. Respetar las líneas de encabezado (directivas para el ensamblador). Las definiciones contenidas en los archivos header (.h) de las librerías de hardware (HAL) pueden ser modificadas para adaptarlas a las necesidades de conexión de los GPIO de cada trabajo.
- En el PDF "*BCM2835-ARM-Peripherals*" se detalla el funcionamiento del bloque GPIO y ARM Timer. Identificar las direcciones de los registros necesarios para la configuración y control de los bloques. Importante: hay un error en la dirección base de los registros del GPIO, esto quiere decir que se deben ignorar las direcciones que figuran y deben ser reemplazadas por:
    - `Peripheral Base Address = 0x3F000000`
    - `GPIO Offset Address = 0x200000`

De esta forma la dirección del primer registro del bloque de los GPIO (correspondiente al registro `GPFSEL0`) está dada por:

- `Peripheral Base Address + GPIO Offset Address = 0x3F200000`.

- El programa debe habilitar el/los puertos a utilizar y configurarlos como salida (en el caso del parlante) y como entrada (para el caso del pulsador SW). Para ello se debe identificar los campos **`FSELn`** (donde **"n"** corresponde al número del GPIO que se desea configurar) de 3 bits c/u, sabiendo qué : 000 = entrada y 001 = salida. Notar que cada registro **`GPFSELx`** permite configurar 10 puertos (`GPFSEL0` del 0 al 9, `GPFSEL1` del 10 al 19…).

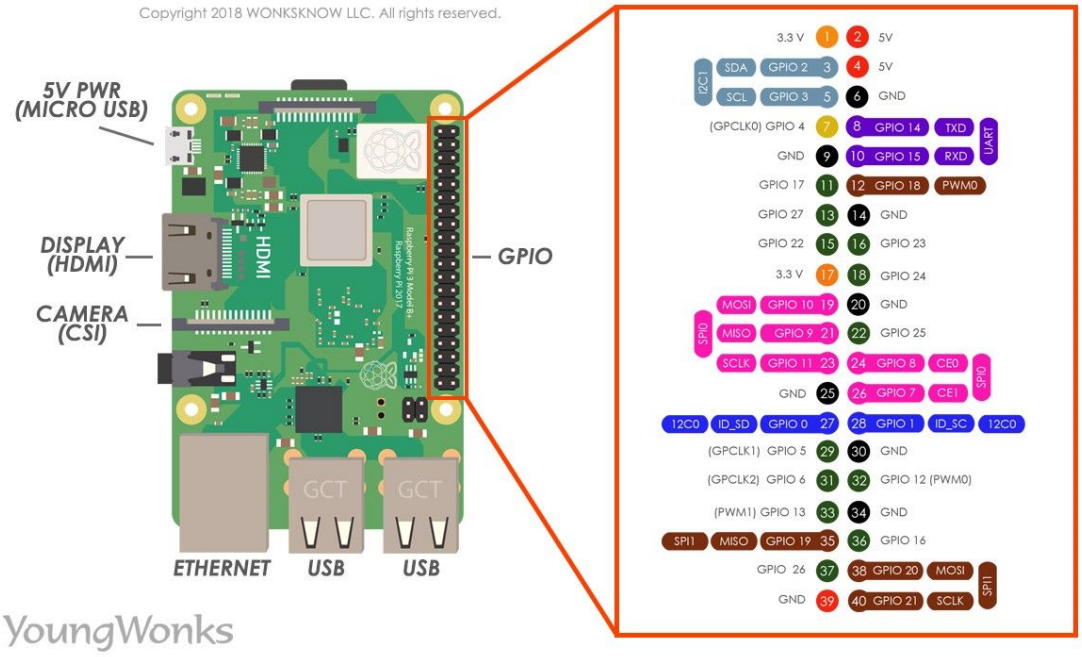

Fig 3.Raspberry Pi 3 pinout

- Poner “1” en una salida: utilizar el campo **`SETn`**, dentro del registro **`GPSETx`**, colocando 1 en los bits correspondientes a los GPIO que se desea encender.
- Poner “0” en una salida: utilizar el campo **`CLRn`**, dentro del registro **`GPCLRx`**, colocando 1 en los bits correspondientes a los GPIO que se desea apagar.
- Consultar el estado de una entrada para evaluar el SW: utilizar el campo **`LEVn`**, dentro del registro **`GPLEVx`**. Se debe leer el contenido del registro correspondiente y evaluar el contenido del bit LEVn de interés. De esta forma:
    - Si el bit LEVn = “0” → SW pulsado.
    - Si el bit LEVn = “1” → SW suelto.
- Crear un **bucle infinito** para que el programa se ejecute mientras la Raspberry Pi permanezca encendida.
- Dejar una **línea vacía** al final del código. El toolchain espera esta línea vacía para asegurarse de que el archivo realmente terminó. Si no se coloca, aparece una advertencia cuando se corre el ensamblador.

Para crear la imagen

Abrir el terminal de la computadora y dirigirse al directorio de la carpeta “template”, escribir el comando `make` y presionar enter. Si no ocurre ningún error, se generan varios archivos, siendo los principales: `main.list`, `main.o` y `kernel7.img`.

Para correr el programa en la Raspberry Pi

- Copiar el archivo `kernel7.img` en la SD, reemplazando la imagen actual.
- Poner la SD en la Raspberry.
- Conectar los módulos periféricos en los puertos de GPIO correspondientes.
- Alimentar la placa desde el USB con el cargador de celular.