FACULTAD DE INGENIERÍA | UNIVERSIDAD NACIONAL DE LA PLATA

Sistemas embebidos:

Trabajo final
Analizador de espectro

Alumno: Tomás Ferreyra.
Sistemas embebidos. Departamento de Electrotecnia. FI, UNLP

3 de agosto de 2024

1. Introducción

En el siguiente informe se detallan los procedimientos, pruebas y herramientas necesarias para llevar a cabo el diseño e implementación de un analizador de espectro digital impulsado por un microcontrolador y demás componentes. El sistema debe ser capaz no solamente de analizar el espectro en frecuencia de cualquier señal sino que también debe poder generar una sinusoide que varíe su amplitud y frecuencia a modo de testeo para el correcto funcionamiento del equipo.

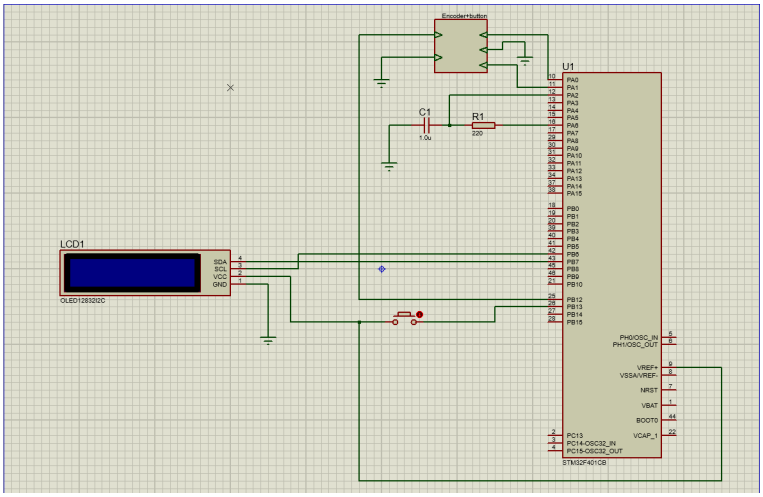
2. Desarrollo

2.1. Diagrama de conexiones y componentes utilizados

Antes de explicar el funcionamiento del equipo se detallaran los componentes que se utilizaron y como fueron conectados los mismos:

- Microcontrolador STM32F103C8T6.
- Resistencia de 270Ω.
- Capacitor electrolitico de 1μF.
- Encoder + pulsador.
- Pulsador.
- Pantalla OLED.
- Modulo conversor UART USB.

Figura 1



2.2. Funcionamiento del equipo

El dispositivo está impulsado por un STM32F103C8T6, que es un microcontrolador de 32 bits de la familia STM32F1 de STMicroelectronics que se basa en el núcleo ARM Cortex-M3. Este fue programado para generar señales senoidales que varíen en amplitud y frecuencia, para leer señales analógicas de entrada con el objetivo de luego realizar la transformada rápida de Fourier y así poder mostrarlas en pantalla con un menú para varias opciones. El microcontrolador también fue programado para exportar el espectro de frecuencia por UART.

Debido que el microcontrolador carece de un conversor digital-analógico, fue necesario generar el seno a través de una portadora modulada por ancho de pulso (PWM) y un filtro basa-bajos. Aquí se toman ciento treinta muestras de un periodo de un seno para luego modular la PWM con estos valores. Para lograr esto, es necesario que la PWM sea ciento treinta veces más rápida que el seno, lo que nos simplifica la eliminación de la componente frecuencial de esta portadora simplemente aplicando un filtro pasa-bajos. El seno debe poder variar su frecuencia con una máxima de 500Hz debido al muestreo que será explicado más adelante, y una frecuencia mínima, para aprovechar la pantalla de 128 pixeles, de 4Hz, ya que 500HZ/128=3.9Hz. El filtro fue realizado con un capacitor electrolítico de 1μF y una resistencia de 270Ω que provoca una frecuencia de corte de 590Hz. La PWM fue generada con el TIMER 3 del microcontrolador.

Por otro lado, fue configurado y programado el conversor analógico-digital (ADC) del STM32 para la adquisición de datos con una frecuencia de 1kHz. Debido al teorema de Nyquist la frecuencia máxima que puede muestrear el ADC es de 500Hz. Luego de capturar 256 muestras se realiza la transformada rápida de Fourier y se imprime, en tiempo real, en la pantalla OLED, la gráfica de esta cuenta. También está la opción de enviar por UART las muestras instantáneas cada vez que se presiona el pulsador de UART.

2.3. Implementación

Para realizar el proyecto fue necesario utilizar FreeRTOS, que es es un sistema operativo en tiempo real de código abierto que es ampliamente utilizado en sistemas embebidos. Gracias a este sistema operativo se pudo trabajar con tareas que permitieron el desarrollo del código necesario para la programación del microcontrolador. Se utilizaron cuatro tareas sincronizadas con la interrupción del TIMER 2, que se utilizo para tomar las 256 muestras cada 1ms.

Las tareas fueron las siguientes:

- SignalGeneratorTask: Utilizada para la generación de la señal PWM con frecuencia variable que modula el seno. Aquí se usa el TIMER 3 y el canal 6 del DMA 1, con el objetivo de no ocupar el procesador con la generación de la señal.

- InputControlerTask: Tarea que realiza un monitoreo de las entradas utilizadas en el proyecto, que fueron dos pulsadores y un encoder. Tanto el encoder como el primer pulsador fue empleado para variar parámetros y como interfaz de usuario, mientras que el otro pulsador solo fue usado para imprimir por UART.
- ScreenControlerTask: Aquí se utiliza una máquina de estados, que será explicada posteriormente, que se encarga del cambio de las pantallas según lo que el operador decida.
- DataControlerTask: Esta tarea se encarga de realizar la transformada rápida de Fourier con las 256 muestras tomadas.

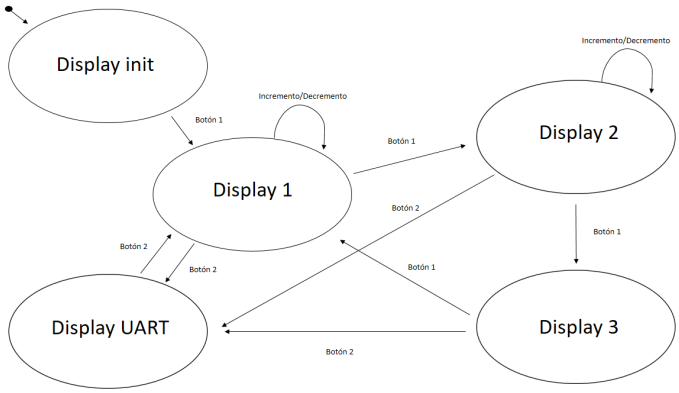
Los mecanismos de comunicación entre tareas fueron los siguientes:

- Notificación: La tarea DataControlerTask notifica a la tarea ScreenControlerTask cada vez que esta tiene un arreglo con la FFT de una señal muestreada.
- Cola FIFO: Una cola FIFO, por sus siglas en inglés: "primero entra, primero sale", es una estructura de datos en la que los elementos se almacenan en el orden en que se insertan, y se retiran en el mismo orden. La tarea InputControlerTask genera distintos eventos según que entrada se accione y luego los mete en una cola FIFO, para ser leído por la tarea ScreenControlerTask y así saber que eventos recibe la máquina de estados.

2.4. Máquina de estados de la interfaz de usuario

Para las acciones llevadas en la pantalla OLED se realizó una máquina de estados que realiza los cambios necesarias en cada pantalla según las entradas que se activen. Se comienza por "Display Init"que da una bienvenida al dispositivo haciendo una y mediante el evento de apretar el botón del encoder se sucede a la pantalla principal, "Display 1". Aquí se muestra la FFT de la señal de entrada, el cursor y, la frecuencia y tensión que marca el cursor. Luego con el botón del encoder se pasa a la segunda pantalla, "Display 2", que aquí el usuario elige se desea modificar el cursor de la primera pantalla, la amplitud o la frecuencia de la señal de salida. Luego presionando nuevamente el botón se sucede a la tercera pantalla o "Display 3", que muestra los valores de tensión y frecuencia de la señal de entrada y el cursor. Por otro lado, desde cualquiera de los estados "Display 1", "Display 2.º "Display 3", si sucede el evento de que se presiona el segundo botón, se pasa al estado "Display UART"que envía por UART mediante DMA la grafica de la FFT de la señal de entrada, una vez terminada esta acción se vuelve al estado "Display 1".

Figura 2



3. Validación

3.1. Sinusoide y PWM

A continuación se muestran las capturas del osciloscopio de la entrada y salida del filtro, y de la señal a alta, media y baja frecuencia.

Figura 3

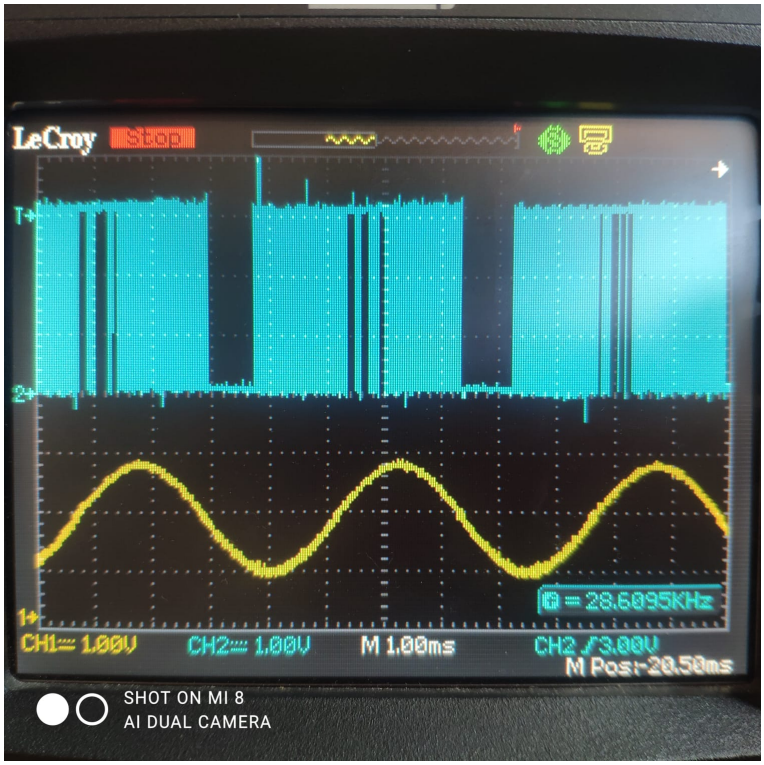


Figura 4

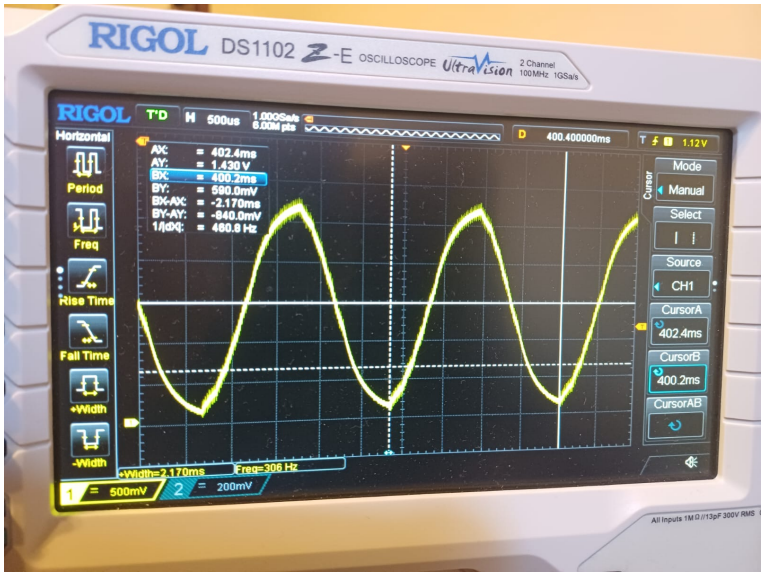


Figura 5

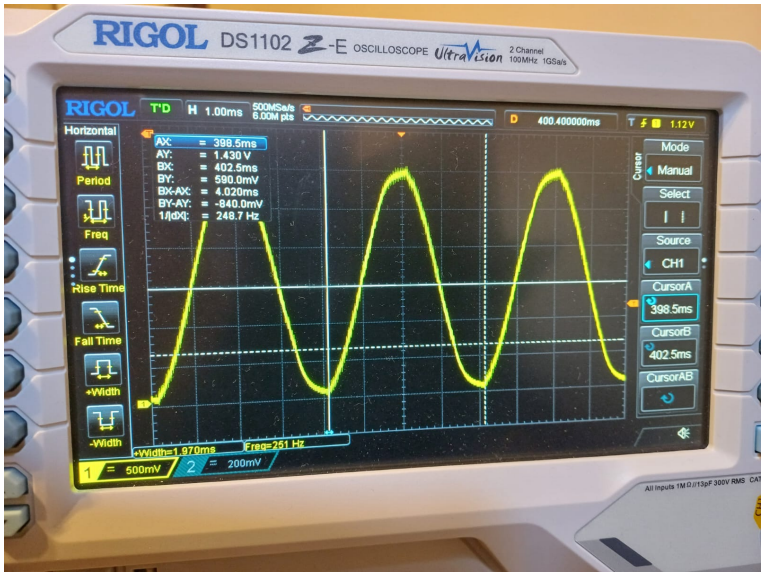
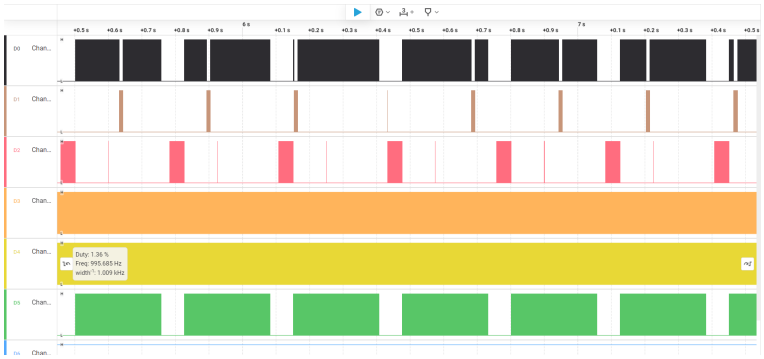


Figura 6

3.2. Análisis de los trace hooks

Los trace hooks en FreeRTOS son macros de callback que se ejecutan ante eventos específicos del RTOS. En este proyecto, fueron utilizados estos hooks para monitorear cuándo se llama a una tarea y si se ejecuta la tarea IDLE. Esto se logró redefiniendo los macros $traceTASK_SWITCHED_IN()$ y $traceTASK_SWITCHED_OUT()$ para invocar funciones que manipulan pines GPIO. Luego, se asignó un valor único a cada tarea usando el campo pxTaskTag del TCB, permitiendo identificar visualmente cuál tarea está en ejecución. A continuación, se muestran los trace hooks que se capturaron.

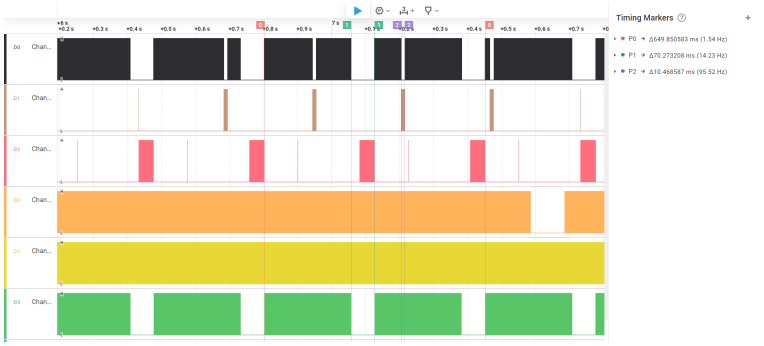
Figura 7



- color negro: Tarea IDLE.
- color marrón: Tarea SignalGeneratorTask.
- color rosa: Tarea ScreenControlerTask.
- color naranja: Tarea InputControlerTask.
- color amarillo: Tarea DataControlerTask.
- color verde: Callback del TIMER 2.

Lo primero que analizaremos de aquí es el tiempo de ejecución de la tarea IDLE. Para ello estimaremos cuanto tiempo se ejecuta en un periodo de tiempo determinado.

Figura 8



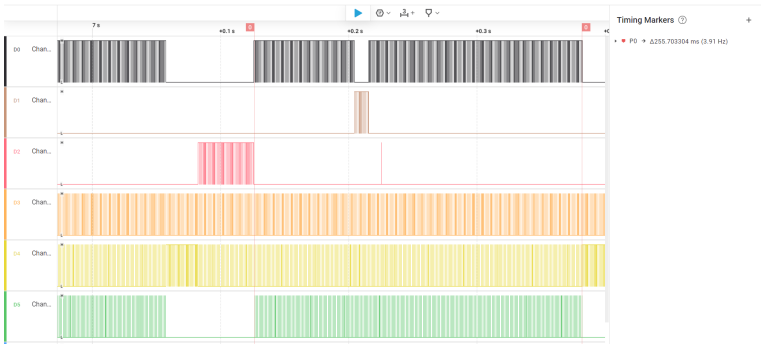
En el tiempo tomado de aproximadamente 650ms, está en bajo al rededor de 160ms, lo que sugiere que el factor de utilización es:

$$U = \frac{160ms}{650ms} = 25$$

Esto quiere decir que se está supera ampliamente la condición de Rate Monotonic del 69 %, por lo tanto fueron asignadas las mayores prioridades a las tareas que tienen menor periodo.

Lo siguiente a analizar seria que se este cumpliendo las 256 muestras tomadas en el callback del TIMER 2.

Figura 9

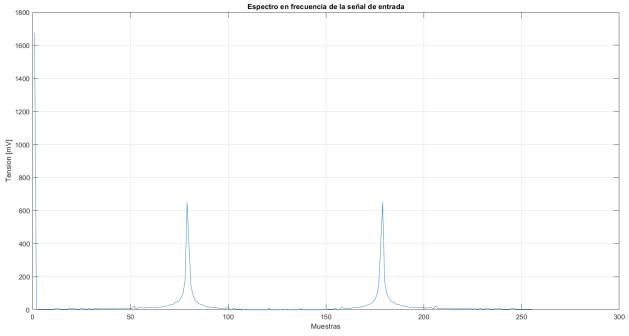


Como se ve en la imagen anterior, se toman muestras en un lapso temporal de 256ms cada 1ms lo que comprueba la correcta temporalización de esta actividad. Aquí también se puede ver el "hueco"temporal que se forma cada vez que recoge las 256 muestras y que acto seguido a esto se ejecuta la taea DataControlerTask con más frecuencia. Es en este momento que la tarea recibió la orden de que las muestras ya están completas y que están listas para realizar la FFT. Luego se ve como se abre paso a la ejecución la tarea ScreenControlerTask para graficar el espectro en frecuencia de la señal muestreada. En cuanto a la tarea SignalGeratorTask, esta muestra fue tomada a propósito en un momento en el que se le estaba cambiando el periodo a la señal generada, por lo que se muestra bastante activa. Por otro lado, se ve como la tarea InputControlerTask sigue ejecutándose permanentemente.

3.3. Envío de la gráfica por UART

A continuación se muestra la gráfica del espectro en frecuencia de una señal senoidal de frecuencia media enviada por UART, y luego representadas las muestras en MatLab.

Figura 10



3.4. Heap y Stack de cada tarea

Un aspecto muy importante a tener en cuenta en este proyecto fue el espacio de memoria tanto del heap del sistema operativo, como el stack de cada tarea. A continuación se ven capturas de pantalla realizadas en tiempo de ejecución luego de pasar por los lugares más críticos del código.

Figura 11

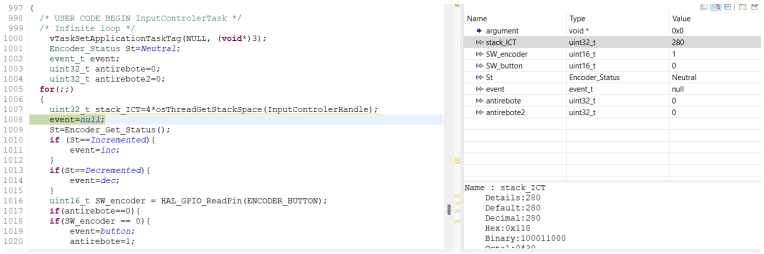


Figura 12

Figura 13

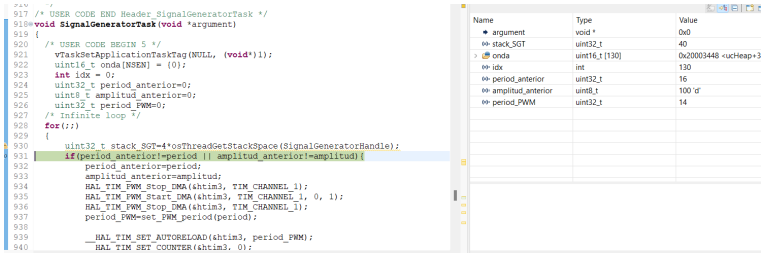
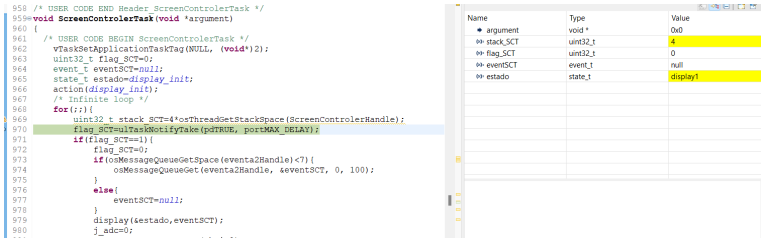


Figura 14



En las anteriores imágenes, se puede ver en las variables $stack_SGT, stack_SCT, stack_ICT, stack_DCTyfreeHeap$ que todas son mayores a cero, lo que significa que sobró espacio de memoria de cada tarea y del heap del sistema.