ARBITRARY WAVEFORM GENERATOR WITH RPI PICO*

*

1st Agustín Marcelo Zuliani
Estudiante de Ingeniería Electrónica
FCEIA
Rosario, Argentina
amzuliani02@gmail.com

2nd Daniel Márquez
Departamento de Sistemas e Informática (DSI)
FCEIA
Rosario, Argentina
dmarquez@fceia.unr.edu.ar

***Abstract*—Este paper trata sobre la implementación de un generador de formas de onda arbitrarias basado en la placa de desarrollo Rasbperry Pi Pico, el cual es un microcontrolador que posee un periférico particular, denominado PIO (Programmable Input/Output) que junto con el módulo DMA permiten generar la señal analógica de salida con un conversor digital a analógico (DAC) tipo escalera R2R. Este trabajo fue realizado como parte de prácticas finales de diferentes materias que abarcaban diseño electrónico y sistemas embebidos con programación basada en sistemas operativos en tiempo real (RTOS).**

I. Introducción

Un generador de formas de onda arbitrarias (AWG) permite generar funciones periódicas cargando puntos en un buffer y reproduciéndolos mediante un DAC a la frecuencia solicitada. La resolución depende de la profundidad del buffer y del DAC, afectando la frecuencia máxima alcanzable. Mayor resolución implicará menor frecuencia máxima.

El objetivo del proyecto fue construir un AWG de bajo costo para ensayos electrónicos, capaz de generar señales de hasta 1 MHz y 10 Vpp, con control digital completo, ajuste de offset y visualización en pantalla en tiempo real. Además, se buscó que fuera escalable para futuras mejoras en amplitud, frecuencia y control. La principal dificultad fue hallar un microcontrolador con suficiente velocidad de muestreo para enviar datos al buffer sin interrupciones. Inicialmente se evaluó el ESP32 con DAC interno y se realizaron pruebas con una escalera R2R, pero ninguna opción alcanzó las especificaciones requeridas. La llegada de la Raspberry Pi Pico en 2022 resolvió esta limitación gracias a su periférico PIO, capaz de controlar GPIOs a alta velocidad de manera independiente a los núcleos principales.

El desarrollo inició como trabajo práctico de Dispositivos y Circuitos Electrónicos IV, comenzando con una salida R2R simple y el diseño de una PCB experimental para verificar el funcionamiento básico. Posteriormente, en la materia Sistemas Embebidos Avanzados II, se incorporaron FreeRTOS y una máquina de estados implementada con Quantum Leaps. Esto permitió organizar el firmware en tareas paralelas y gestionar de forma estructurada la configuración de señales, logrando un sistema escalable, estable y funcional luego de varias etapas de pruebas y ajustes. Además se realizó el diseño final de la PCB enviado a fabricar a un proveedor de China.

II. Alcance del proyecto

Una de las principales cuestiones al momento de realizar un proyecto es el alcance del mismo, de esta forma podemos medir si los requisitos solicitados fueron cumplidos. Para este proyecto se nombran los siguientes:

- Rangos de frecuencia: $f_{min} = 1$ Hz y $f_{max} = 1$ MHz.
- Resolución de frecuencia: $f_{res} = 1$ Hz.
- Alimentación: $V_{in} = \pm 15$ V
- Tensión de salida: $V_{out} = 10$ Vpp.
- Corriente máxima de salida: $I_{max} = 100$ mA.
- Impedancia de salida: $Z_{out} = 0\ \Omega$ y $Z_{out} = 50\ \Omega$.
- Ajuste de offset.
- Visualización por pantalla HMI.

III. Diseño del Hardware

A. Conversión digital a analógica con escalera R2R

Los GPIOs de un microcontrolador solo pueden presentar dos estados lógicos, por lo que fue necesario implementar un método que permitiera generar una señal analógica. Aunque se evaluaron DACs con interfaz SPI, estos no cumplían con la frecuencia máxima requerida ni con el presupuesto disponible. La alternativa más viable fue utilizar una escalera R2R controlada por la Raspberry Pi Pico.

El procedimiento consiste en almacenar cada punto de la señal en un buffer de memoria y transferirlo mediante DMA al periférico PIO, que se encarga de habilitar los GPIOs correspondientes de acuerdo con cada byte enviado. Este método permite alcanzar frecuencias elevadas al trabajar de manera independiente del procesador principal, logrando una conversión digital-analógica eficiente con un hardware mínimo.

B. Etapa de amplificación y desacoplamiento DC

Una vez generada la señal de salida del R2R, la señal entra en la etapa de amplificación. Para esto utilizamos un AO en configuración inversora y para generar una ganancia de forma digital usamos un potenciómetro digital que tiene un rango de $0-10$ kΩ con una resolución de $100\ \Omega$. La elección del AO no fue sencilla debido a las limitaciones en frecuencia. Como resultado de investigación se eligió el LM6172, el cuál puede verse la hoja de datos en [1].

Como la señal de salida (tanto del R2R como de la etapa de amplificación) posee un nivel de continua, se incluyó una etapa de desacoplamiento DC que adiciona a la señal un valor en DC opuesto al que presenta la señal amplificada.

C. Etapa de Offset y suma

Para realizar la etapa de offset, se utilizó un LM358 (ver [2]) para generar dos filtros pasa bajos uno para offset positivo y otro negativo, ambos con $V_{offMax} = |5|$ Vpp. Por último todas estas señal mencionadas ingresan a un mezclador (sumador) inversor que posee una amplificación fija.

IV. Diseño del Software

El firmware se desarrolló con tres objetivos principales: claridad, legibilidad y escalabilidad para futuras mejoras. Para lograrlos se adoptaron tres enfoques clave:

- **Máquina de estados**: Permitió organizar la navegación entre menús de configuración de manera simple y flexible, facilitando la incorporación de submenús o nuevas funciones.
- **RTOS**: Se utilizó FreeRTOS para ejecutar tareas en paralelo, como la lectura de pulsadores, la detección de cambios en el encoder y la actualización de la pantalla, garantizando un funcionamiento fluido sin interrupciones.
- **Programación orientada a objetos (POO)**: Cada elemento del hardware se representó como un objeto (potenciómetro digital, etapa de amplificación, offset, etc.), lo que facilitó la integración de nuevas funciones y el mantenimiento del código.

Estos tres pilares dieron lugar a un firmware modular y escalable, adecuado para futuras ampliaciones del sistema.

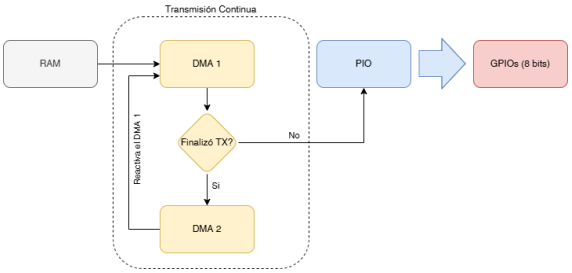

Fig. 1. Diagrama de bloques de transferencia del buffer en RAM al periférico PIO

A. Transferencia de información

La comunicación entre la RAM y el PIO se basa en un flujo continuo de datos. Cada punto de la señal se almacena en un buffer en RAM y se transfiere al periférico PIO mediante DMA, que carga en paralelo los GPIOs correspondientes. Para mantener la continuidad, se emplean dos DMAs encadenados: el primero realiza la transferencia y, al finalizar, activa el segundo, que reinicia al primero en un ciclo constante, ver figura 1. Una posible mejora sería implementar un modo ping-pong con dos buffers, reduciendo la latencia a costa de mayor complejidad lógica y uso de memoria.

B. Máquina de estados

Quantum Leaps es un framework que permite generar gráficamente máquinas de estados basadas en eventos mediante el uso de objetos activos que contienen atributos, clases y una SM (state machine) asociada. La máquina de estados en forma de menús permite que el usuario avance o retroceda, configurando los parámetros principales del AWG. Los estados más relevantes de la máquina de estados son:

- Tipo_Func: selecciona el tipo de señal a generar.
- Freq: ajusta la frecuencia de salida.
- Amplitud: configura la amplitud de la señal.
- Offset: define el nivel de offset.
- Confirm_Config: confirma y aplica los cambios realizados.

Dichos cambios producidos por el usuario se reflejan en la pantalla Nextion de 2.4’ manipulada a través de UART. Se puede ver la MEF completa en [3].

C. Sistema Operativo en Tiempo Real

Para la gestión del firmware se utilizó FreeRTOS, elegido por ser de código abierto y capaz de integrarse con Quantum Leaps. La máquina de estado se ejecuta como una tarea estática, mientras que en paralelo corren otras tareas esenciales: detección de pulsadores, manejo de interrupciones del encoder y generación de la señal (AWG) que son dinámicas.

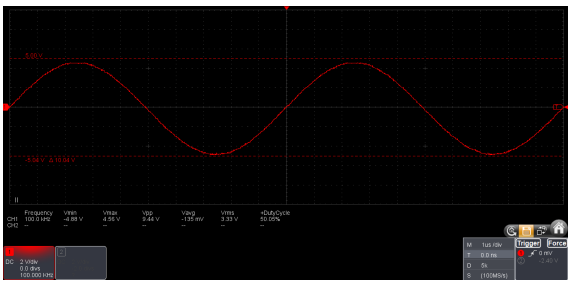

Fig. 2. Señal senoidal a 100 KHz con 10 Vpp.

D. Programación orientada a objetos

Una vez validada la base funcional, se reescribió el software bajo un enfoque de programación orientada a objetos (POO). Cada componente del hardware (potenciómetro digital, etapa de amplificación, offset, desacoplamiento de continua y generación de señal) fue modelado como un objeto independiente. Estos objetos se integraron en un objeto principal denominado señal, que representa de forma unificada tanto la lógica de software (generación, buffer y configuraciones) como el hardware asociado. Este enfoque, aunque laborioso, permitió un código modular, escalable y fácil de mantener.

V. Resultados

A. Ensayos finales

Una vez realizada toda la programación y hardware preliminares se realizaron mucho ensayos para que el proyecto anduviera de manera correcta. El resultado final puede verse en la figura 2 donde se muestra una señal senoidal a 100 KHz y 10 Vpp sin offset.

Puede verse mas imágenes de ensayos directamente en el repositorio del proyecto en [3]. Entonces . . . *¿cumplimos con todos los objetivos del proyecto?* Podría decirse que en su mayoría si, por ejemplo: en la señal cuadrada llegamos hasta 200 KHz con buena calidad, en la senoidal hasta 1.5 MHz y la triangular hasta 300 KHz. Cumplir esas condiciones sin usar un FPGA y para un proyecto de dos materias es un rotundo éxito. Sin embargo siempre hay cosas que pueden mejorarse tanto a nivel de software, hardware como de también de costos.

B. Futuras mejoras

Se identificaron varias mejoras potenciales para optimizar el sistema:

- Costos: Eliminar la pantalla TFT y reemplazarla por un programa de control desde PC o dispositivos móviles, lo que reduciría significativamente el costo del sistema.
- Etapa de potencia: Completar el diseño de una etapa de potencia que permita proveer impedancia controlada y mayor corriente de salida.
- Potenciómetro digital: Sustituirlo por uno capaz de soportar tensiones mayores a ±5 V, lo que permitiría alcanzar amplitudes de hasta 12 Vpp, típicas en generadores comerciales.
- Robustez de hardware: Mejorar la tolerancia a perturbaciones externas e incluir una etapa de alimentación más estable.
- Control cerrado: Incorporar lazos de control para ajustar parámetros en tiempo real.
- Software: Corregir detalles en la detención de la señal, optimizar la elección de frecuencias PWM en función del diseño de PCB y agregar control inalámbrico (Bluetooth o WiFi) para operar el sistema desde PC o dispositivos móviles.

VI. Licencias

El trabajo posee licencias no comerciales de Qp y libres de FreeRTOS. Por lo que decidió que el proyecto sea compartido bajo las siguientes condiciones [4].

References

[1] Texas Instruments, *Lm6172 dual, high-speed, low-power, low-distortion, voltage-feedback amplifier*, Datasheet revisado en diciembre de 2024, Texas Instruments, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm6172.pdf.

[2] Texas Instruments, *Lm358 low-power dual operational amplifier*, Datasheet revisado en 2023, Texas Instruments, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm358.pdf.

[3] A. Zuliani. “Awg rpi pico.” (2025), [Online]. Available: https : / / github . com / Agustin586 / AWG - RP2040/tree/main.

[4] A. Zuliani, *Awg rpi pico © 2025 de agustín zuliani, con licencia creative commons atribución-nocomercial 4.0 internacional. para ver una copia de esta licencia, visite https://creativecommons.org/licenses/by-nc/4.0/*.