https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146

Vol. 4, Núm. 2. abril-junio 2026, págs. 124 - 147

Propuesta de un sistema de adquisición y generación de señales en plataformas Raspberry Pi

Signal acquisition and generation system on Raspberry Pi platforms

Información del reporte:

Para citar este reporte técnico:

Valera Orozco, B., Ruiz Botello, G.A., y Roldán Serrato, K.L. (2026). Propuesta de un sistema de adquisición y generación de señales en plataformas Raspberry Pi. *Cuadernos Técnicos Universitarios de la DGTIC, 4*(2), páginas (124 - 147). https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146

Benjamín Valera Orozco
Instituto de Ciencias Aplicadas y Tecnología
Universidad Nacional Autónoma de México
benjamin.valera@icat.unam.mx
ORCID: 0009-0006-7730-8750

Gerardo Antonio Ruiz Botello
Instituto de Ciencias Aplicadas y Tecnología
Universidad Nacional Autónoma de México
gerardo.ruiz@icat.unam.mx
ORCID: 0009-0000-3525-4252

Karen Lucero Roldán Serrato
Instituto de Ciencias Aplicadas y Tecnología
Universidad Nacional Autónoma de México
lucero.roldan@icat.unam.mx
ORCID: 0009-0006-9761-5029

Resumen

En la actualidad, el mercado de los sistemas operativos ha tomado un rumbo hacia el uso de sistemas abiertos, libres y consolidados entre la comunidad de especialistas en cómputo y electrónica. Un caso aplicado es la computadora de placa única Raspberry Pi, a la cual se le pueden instalar diferentes versiones de sistema operativo de plataforma abierta y gratuita. Las aplicaciones enfocadas en la electrónica encuentran un gran potencial en la Raspberry Pi debido a su capacidad de desarrollo de sistemas completos gracias a su conector de propósito general de 40 pines. En este proyecto, se diseñó e implementó un sistema electrónico para la adquisición y generación de señales eléctricas basadas en una microcomputadora

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 125 - 147

Raspberry Pi y un sistema operativo Raspbian OS. Esta herramienta resultó una implementación adecuada y útil ya que existe un gran número de aplicaciones de instrumentación en las que se requiere este tipo de procesos de manipulación de señales. La aportación de este trabajo fue el desarrollo de un sistema a bajo costo y optimizado para la adquisición y generación de señales, el cual estuvo creado con programación en Python y un ambiente gráfico de visualización. Éste fue utilizado en aplicaciones concretas como laboratorios de electrónica o de experimentación en telecomunicaciones.

Palabras clave: Sistema de adquisición de datos, procesamiento de señales, Raspberry Pi 5, instrumentación electrónica.

Abstract

Currently, the operating systems market has been shifting toward open, free, and community-established solutions widely adopted by computing and electronics specialists. A practical example is the Raspberry Pi single-board computer, which supports various versions of open and free operating systems. Applications in electronics find significant potential in the Raspberry Pi due to its ability to support full system development through its 40-pin general-purpose connector. In this project, an electronic system for the acquisition and generation of electrical signals was designed and implemented based on a Raspberry Pi microcomputer running Raspbian OS. This tool proved to be an appropriate and useful solution, given the wide range of instrumentation applications that require signal processing capabilities. The main contribution of this work is the development of a low-cost, optimized system for signal acquisition and generation, implemented using Python programming and a graphical visualization environment. It was applied in specific contexts such as electronics laboratories and telecommunications experimentation.

Keywords: *Data acquisition system, signal processing, Raspberry Pi 5, electronic instrumentation.*

1. INTRODUCCIÓN

Hacia el año 2014, comenzaron a desarrollarse diversas aplicaciones con base en Raspberry Pi orientadas no sólo al ámbito educativo, sino también a aplicaciones científicas e industriales (Sreejith *et al.* 2014). Estas capacidades extendidas incluyen dispositivos que compiten con grandes ventajas frente a los microcontroladores Arduino. Por ejemplo, en el ámbito electrónico, además de incluir las características tradicionales en los microcontroladores Arduino, como entradas y salidas de propósito general, la plataforma Raspberry Pi incorpora herramientas para manipular motores, luces, sensores y más. Sin embargo, la principal ventaja de la plataforma Raspberry Pi es que cuenta con un sistema operativo comparable a Windows, con la capacidad de gestionar dispositivos periféricos complejos y bases de datos en ambientes de programación como Python y C.

En este ambiente de crecimiento, se planteó la posibilidad de incursionar en el uso de herramientas modernas para el desarrollo de dispositivos electrónicos capaces de generar y capturar señales analógicas en la plataforma Raspberry Pi. La principal motivación es disminuir la dependencia tecnológica actual de los sistemas de cómputo basados en microprocesadores Intel y en el sistema operativo Microsoft. Adicionalmente, los procesos de conversión analógica a digital (ADC) y digital a analógica (DAC) (ver glosario en el Anexo A) son fundamentales en la elaboración de instrumentación para aplicaciones industriales y de diversos ámbitos de la actividad humana.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 126 - 147

Conviene señalar los desarrollos de instrumentación en el área de cómputo aplicado en plataformas de propósito específico. En 2014, algunas plataformas del Internet de las Cosas (IoT) incluyeron el uso de convertidores analógico-digitales externos, tales como el MCP3008, ADS1115 o el de alta precisión ADS1263, los cuales permiten digitalizar señales a través de sensores y sistemas analógicos (Medina, 2017; Rodríguez, 2018). Estos dispositivos son ampliamente utilizados por su fácil adaptación con interfaces como SPI e I2C, que permiten una integración directa con la Raspberry Pi (AG Electrónica, 2024; Meena *et al*., 2019). En 2015 ya se desarrollaban sistemas del IoT enfocados en la adquisición de señales a bajas frecuencias (VLF), los cuales consistían en un receptor VLF, un preamplificador y un convertidor analógico digital sobre una tarjeta Raspberry Pi portable (Gunawan *et al.,* 2020). Así, a partir del 2018 y para robustecer la instrumentación, se incorporaron diversos módulos de validación y calibración a los sistemas de adquisición y medición basados en tarjetas de procesamiento. Estos desarrollos fueron aplicados a equipos industriales como módulos adicionales (Arif *et al*., 2018).

En años posteriores, cobraron relevancia los instrumentos de generación de señales con base en radiofrecuencias FM y aplicadas a tarjetas como Raspberry Pi en sus últimas versiones de ese tiempo. A pesar de que siempre se debe contemplar la compatibilidad de controladores y bibliotecas para los últimos modelos de tarjetas, esta plataforma demostró ser una versión estable en un analizador de espectros, donde se observaron buenas evidencias en los componentes armónicos por discriminar (Martínez-Quintero *et al*., 2019). Por otra parte, hacia 2020, la literatura reportó el uso de plataformas basadas en *software* libre, como GNU Linux, programación en Python y librerías compatibles e incluidas en tarjetas como Raspberry Pi y Arduino en los modelos de 3, 4 y 5 (Cassel Barbosa *et al*., 2020). En algunas aplicaciones como las del campo biomédico, se han reportado múltiples implementaciones orientadas a la adquisición de señales fisiológicas como EMG (electromiografía) y ECG (electrocardiografía), mediante la combinación de Raspberry Pi con microcontroladores auxiliares que se encargan del preprocesamiento y adquisición de señales a mayor resolución y frecuencia (Chen & Liu, 2025).

Adicionalmente, las tarjetas de propósito específico como las de Adafruit y otras plataformas similares, entre ellas Orange Pi, Banana Pi y Raspberry Pi, permiten implementar diversas aplicaciones de instrumentación, tales como la adquisición y generación de señales (Arif *et al*., 2018; Gunawan *et al*., 2020). Además, se cuenta con la programación en lenguajes como Python y C incluidos en el sistema operativo de la Raspberry Pi, así como la integración de sensores y actuadores (bluemoon16, 2023). En términos de costo y flexibilidad, es una opción ideal para sistemas de adquisición de datos (DAQ) y generación de señales analógicas y digitales (Ellison *et al*., 2024).

De acuerdo con la revisión de los antecedentes, se observa una evolución sostenida en el desarrollo de sistemas de instrumentación basados en plataformas de cómputo de propósito específico, particularmente en el contexto del Internet de las Cosas (IoT). La incorporación de convertidores analógico-digitales externos como MCP3008, ADS1115 y ADS1263 ha permitido la digitalización eficiente de señales provenientes de sensores y sistemas analógicos, facilitando su integración con plataformas como la Raspberry Pi mediante interfaces estándar como I^2C y SPI, soluciones que han destacado por su facilidad de implementación y adaptabilidad en sistemas embebidos de bajo costo. Además, se han desarrollado sistemas de adquisición de señales de baja frecuencia, así como arquitecturas portables basadas en Raspberry Pi que integran etapas de acondicionamiento, amplificación y conversión analógico-digital; estos sistemas han incorporado módulos de validación y calibración que han contribuido a mejorar su robustez y aplicabilidad en entornos industriales como módulos de instrumentación adicionales. De esta manera, se ha extendido el uso de plataformas de generación de señales, incluyendo aplicaciones en

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 127 - 147

radiofrecuencia, donde la Raspberry Pi ha demostrado capacidad para la implementación de analizadores de espectro con resultados satisfactorios en la identificación de componentes armónicos.

A partir de este análisis, se justifica el desarrollo de un sistema de adquisición y generación de señales basado en Raspberry Pi y Python como una solución de instrumentación que responde a la necesidad de plataformas flexibles, de bajo costo y fácilmente reconfigurables que integren en un solo entorno las funciones de adquisición, procesamiento y generación de señales.

La principal aportación de este trabajo es la implementación de una solución integral que aprovecha la arquitectura de la Raspberry Pi junto con Python para desarrollar un sistema DAQ (*Data Acquisition System*) (Anexo A) capaz no sólo de adquirir señales mediante interfaces estándar, sino también de generar señales controladas de manera programable. Esto permite un entorno unificado de instrumentación, reduciendo la dependencia de equipos especializados de alto costo y facilitando la replicabilidad del sistema en aplicaciones académicas, de investigación e incluso industriales de prototipos integrando *software* y *hardware* de código abierto. El sistema propuesto contribuye a la consolidación de plataformas abiertas de instrumentación al integrar adquisición y generación de señales en una misma arquitectura, fortaleciendo la tendencia actual hacia sistemas embebidos modulares, escalables y basados en *software* libre, así como adaptados a ambientes institucionales.

El objetivo del proyecto presentado en este reporte técnico es desarrollar un sistema de adquisición y generación de señales en plataformas Raspberry Pi.

2. DESARROLLO TÉCNICO

Para este proyecto, se eligió Raspberry Pi versión 5 debido a la experiencia previa con esta plataforma, su facilidad de integración y acoplamiento con los elementos de instrumentación y su compatibilidad con otros módulos o sensores externos. Posteriormente, se realizó la implementación de su sistema operativo, Raspberry Pi OS. En cuanto al lenguaje de programación, se empleó Python, debido a su papel fundamental en el manejo y procesamiento de los datos adquiridos. En seguida, se configuró el entorno de desarrollo mediante la instalación de algunas bibliotecas necesarias para la implementación del sistema, entre las que se incluyeron: NumPy, SciPy y Matplotlib, las cuales se utilizaron para el procesamiento, análisis y visualización de señales. Esto exhibió a la Raspberry Pi como una herramienta poderosa para la investigación científica (Pidora, 2024). En cuanto a los protocolos de comunicación y generación de señales, la Raspberry Pi soporta interfaces como I2C, SPI y UART, lo que permite una gran flexibilidad al momento de integrar convertidores analógicos a digital y sensores de distintos tipos. Además, se han documentado múltiples soluciones que incluyen la transmisión y el almacenamiento de datos en la nube, facilitando el monitoreo remoto y el análisis en tiempo real mediante servicios como ThingSpeak, AWS o plataformas personalizadas (Magalhães, 2024).

El desarrollo e implementación de tecnología adaptada a requerimientos electrónicos tiene la desventaja de altos costos de implementación o fallos ante la falta de calibración a largo plazo. Por lo anterior, se propuso la creación de un prototipo moderno en el desarrollo de dispositivos electrónicos para generar y capturar señales analógicas en la plataforma Raspberry Pi. La principal motivación es reducir la dependencia tecnológica actual de los sistemas de cómputo basados en microprocesadores y sistemas operativos propietarios, así como promover el uso de sistemas de instrumentación portables, de menor costo y apoyados en *software* libre para la programación del sistema. Adicionalmente, los

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 128 - 147

procesos de conversión analógica a digital y digital a analógica considerados resultan ser fundamentales en la elaboración de instrumentación para aplicaciones industriales y de diversos ámbitos de la actividad humana.

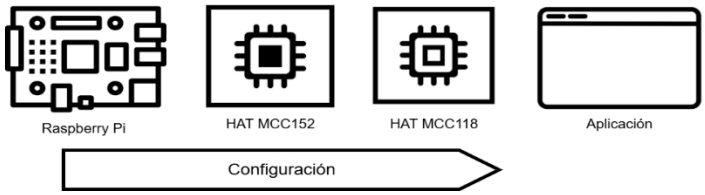
2.1 INTEGRACIÓN DE CIRCUITOS DE SEÑALES CON RASPBERRY PI

El proyecto fue propuesto tomando como base el planteamiento de Llamas (2025), que consiste en integrar un sistema DAQ basado en una computadora de bajo costo y diseño compacto, la Raspberry Pi 5. Este sistema integra dos circuitos HAT (ver glosario en el Anexo A) al conector de 40 pines en la Raspberry Pi, y termina con el desarrollo de un *software* con aplicación real en la generación y adquisición de señales eléctricas. Un circuito HAT tiene el propósito específico de generar señales analógicas a partir de datos digitales en la Raspberry Pi. El otro HAT, en cambio, se encarga de generar datos digitales en la Raspberry Pi a partir de señales analógicas del ambiente exterior. Con estos datos, se pueden procesar parámetros a partir de algoritmos específicos de control o procesamiento digital de señales. Con base en el documento informativo de XOREN Ingeniería (2023), el sistema DAQ desarrollado tiene el enfoque de programación en lenguaje Python, ya que es uno de los lenguajes más populares en la industria por su flexibilidad y solución de problemas sencillos con una sintaxis simple y elegante.

El método incluye la instalación física de los tres circuitos, su configuración en el sistema operativo Raspberry OS y el desarrollo de una aplicación para la adquisición y generación de señales analógicas, como se muestra en la Figura 1.

Figura 1

Método por emplear



La Figura 1 muestra conceptualmente el sistema Raspberry Pi, que hospeda dos subsistemas de tratamiento de señales HAT MCC152 (Measurement Computing, 2019) y HAT MCC118 (Measurement Computing, 2020). El subsistema MCC152 es un circuito electrónico diseñado para la generación de señales de voltaje mediante conversión digital-analógica DAC, a partir de datos digitales, diseñado específicamente para conectarse a la Raspberry Pi. Por otra parte, el subsistema MCC118 es un circuito electrónico para la medición de voltaje ADC, de diseño específico para conectarse también a la tarjeta. El par de circuitos electrónicos MCC152 y MCC118 habilitan a una microcomputadora Raspberry Pi con capacidades extendidas para el procesamiento digital de señales.

En la misma Figura 1, la conexión física entre la Raspberry Pi y los circuitos MCC152 y MCC118 no es suficiente para habilitar las capacidades de procesamiento digital de señales, también se debe configurar el ambiente de programación Python al cargar las utilerías necesarias para este propósito. De esta forma, se puede desarrollar una aplicación funcional que procese datos digitales tanto de entrada como de salida y se pueda interactuar con el entorno real.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 129 - 147

El principal objetivo es implementar un sistema DAQ que pueda ser de utilidad para una aplicación futura, ya sea de control o de procesamiento digital de señales, en una plataforma de cómputo abierta y de bajo costo. Este sistema DAQ puede ser de utilidad para la medición de variables físicas como temperatura, presión, desplazamiento, entre otras. Además, se puede actuar sobre transductores como motores, electroválvulas, resistencias, luces, entre otros.

El resultado esperado es un sistema DAQ con las siguientes características obtenidas a partir de las especificaciones de los circuitos HAT:

Generación de señal analógica
- Excursión analógica: 0 a 5 V.
- Resolución: 12 bits.
- Periodo entre muestras: 12 µs.

Adquisición de señales
- Excursión analógica: 0 a 5 V.
- Tasa de muestreo: 100 kmuestras/s.
- Resolución: 12 bits

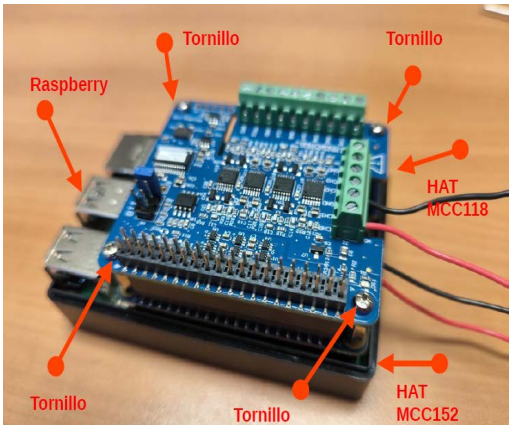
2.2 CONFIGURACIÓN DE HARDWARE DEL SISTEMA ELECTRÓNICO DAQ

El sistema electrónico DAQ adquiere y genera señales analógicas, implementando los procesos DAC y ADC respectivamente. Estos procesos son primordiales en diversas aplicaciones de los sistemas electrónicos para la medición, control y procesamiento de información en sistemas digitales.

Para instalar el primer HAT DAC en la Raspberry Pi, primero se deben colocar los postes de sujeción a la carcasa y el extensor para el conector de 40 pines, como se muestra en la Figura 2. El primer HAT, MCC152, se prepara colocando cables de extensión para el canal analógico 0, se especifica su dirección a cero mediante la ausencia de puente en el conector de direcciones y se especifica el voltaje de operación a 5 V con el puente en el conector de polarización. El primer HAT se conecta a la Raspberry Pi añadiendo 4 postes adicionales y un segundo extensor del conector de 40 pines. El segundo HAT ADC, MCC118, se prepara colocando cables de extensión para el canal 0 y especificando la dirección 1 con un puente en el conector de dirección. Posteriormente, el HAT se monta sobre el arreglo y se sujeta con tornillos. Finalmente, al arreglo se le añade teclado y ratón inalámbricos, pantalla HDMI y polarización.

Figura 2

Sistema DAQ completo



https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 130 - 147

2.3 INTEGRACIÓN DEL SISTEMA

Una vez integrado el *hardware* de la Figura 2, como parte del método empleado, se procede a configurar el *software* para desarrollar una aplicación funcional que efectivamente proporcione tratamiento a las señales de entrada y salida del sistema electrónico. Después, se debe verificar la configuración comprobando el funcionamiento con los programas de prueba que distribuye el fabricante de los HATs. Finalmente, la plataforma estará lista para desarrollar programas propios en aplicaciones específicas.

2.3.1 CONFIGURACIÓN Y APLICACIÓN

En la siguiente descripción de la configuración, se asume que el sistema operativo Raspberry Pi OS está instalado y actualizado en la microcomputadora. El primer paso en la configuración consiste en descargar la *daqhats library* (Measurement Computing Corporation, 2024) en la carpeta raíz del usuario mediante las dos siguientes instrucciones en la terminal de comandos:

```
…/cd ~
…/git clone https://github.com/mccdaq/daqhats.git
```

Posteriormente, se compilan e instalan las herramientas de la librería:

```
…/cd ~/daqhats
…/sudo ./install.sh
```

Mediante este procedimiento, se instala la librería de los HATS y la documentación junto con los archivos de ejemplo en el directorio `/home/usuario/daqhats/`.

2.3.2 VALIDACIÓN Y CALIBRACIÓN

Para verificar la correcta operación de las configuraciones descritas en 2.4.1, se utiliza Thony como ambiente de programación (Jaryd, 2023) en la constitución de un proyecto de aplicación mediante el cual se pueda comprobar la correcta operación de las tarjetas HAT.

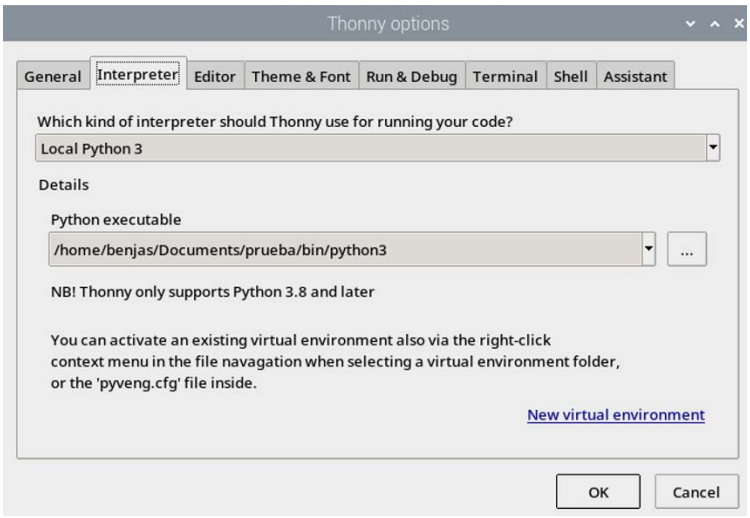
En este sentido y con fines de seguridad computacional, se crea un entorno virtual de programación que aísle la operación de las tarjetas HAT del resto del sistema operativo. Inicialmente, este entorno virtual es utilizado para alojar el primer programa básico de prueba que distribuye el fabricante de los HATS. El programa de prueba es un archivo de Python que genera un voltaje analógico en el canal 0 del HAT MCC152.

Para crear el entorno virtual sobre un directorio vacío de prueba, se selecciona la opción del menú de Thony “Run/Configure interpreter...”, que despliega el diálogo de la Figura 3.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 131 - 147

Figura 3

Diálogo "Thonny options" para crear un nuevo ambiente virtual

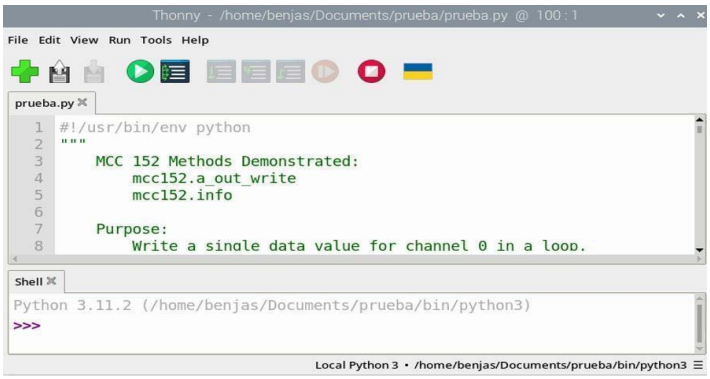


Al presionar el enlace *New virtual environment* de la Figura 3, se despliega una advertencia en donde se hace énfasis que el directorio a seleccionar debe estar vacío para el nuevo ambiente virtual. Al aceptar la advertencia, se selecciona el directorio vacío de prueba y se retorna al diálogo de la Figura 3, en donde se confirma la ubicación del archivo ejecutable.

Al aceptar el diálogo de la Figura 3, se retorna al ambiente de programación Thonny, donde ya es posible probar los ejemplos proporcionados por el fabricante de los HATS con la librería llamada *daqhats*. Con este propósito, se copia el texto del archivo de prueba llamado *analog_output_write.py* del directorio / home/usuario/daqhats/ al documento *untitled* y se salva como *prueba.py*, como se muestra en la Figura 4. Por medio de este programa, es posible generar un voltaje analógico en el canal 0 del HAT MCC152.

Figura 4

Programa de ejemplo en el archivo prueba.py

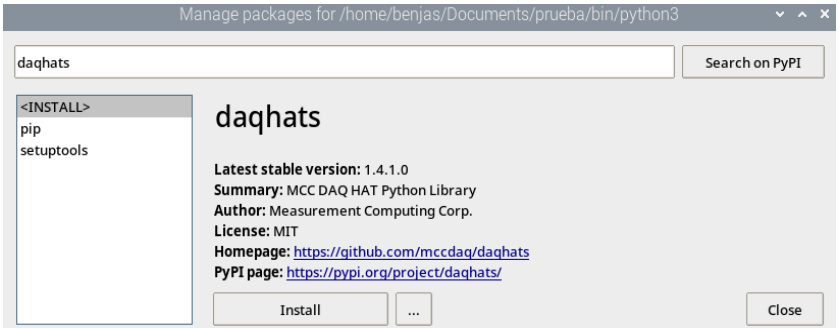


https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 132 - 147

Para ejecutar el programa en *prueba.py*, se debe agregar al ambiente de programación la librería daqhats como un paquete. Para este propósito, mediante la opción *Tools/manage packages...* de Thony, se instala la librería daqhats de la Figura 5.

Figura 5

Instalación de la librería daqhats



El archivo de ejemplo distribuido por el fabricante contiene algunos detalles que deben ser reparados para que el programa se ejecute sin contratiempos. En este sentido, se hacen los siguientes cambios al archivo *prueba.py* y se ejecuta el programa:

Línea 16 se enmascara la instrucción:

```
#from daqhats_utils import select_hat_device
```

Linea 71 se enmascara la instrucción:

```
#address = select_hat_device(HatIDs.MCC_152)
```

Se agrega la línea 72:

```
address=0
```

Entonces, el programa se ejecuta y éste solicita un valor numérico entre 0 y 5 V para generarlo en el canal 0 del HAT MCC152, como se muestra en la Figura 6. Se ingresa un número y se genera el voltaje como se muestra en la Figura 7. En esta situación, se completa la verificación de las configuraciones y se puede crear una aplicación propietaria en Python.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 133 - 147

Figura 6

Programa prueba.py ejecutándose libre de errores

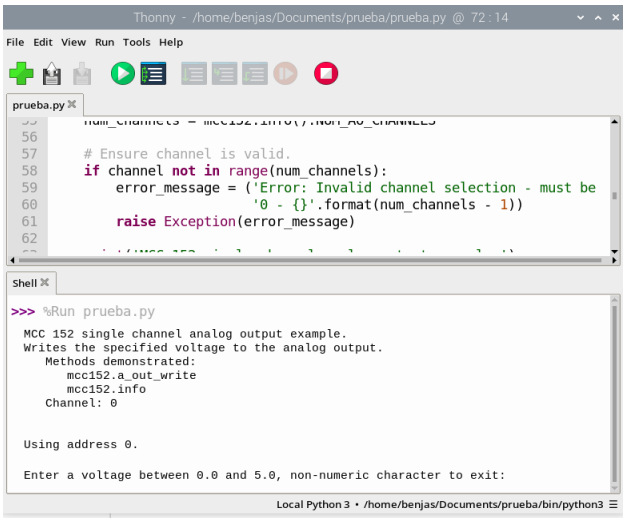
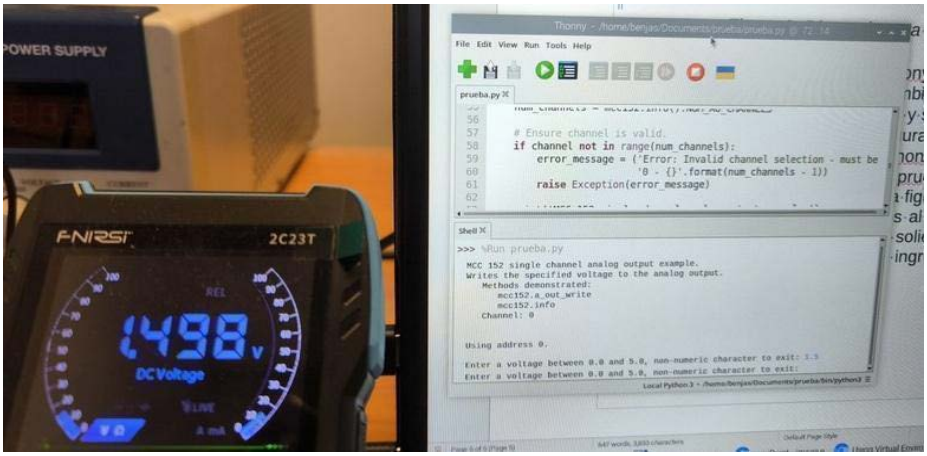


Figura 7

Comprobación del voltaje generado 1.5 V leído con multímetro como 1.498 V



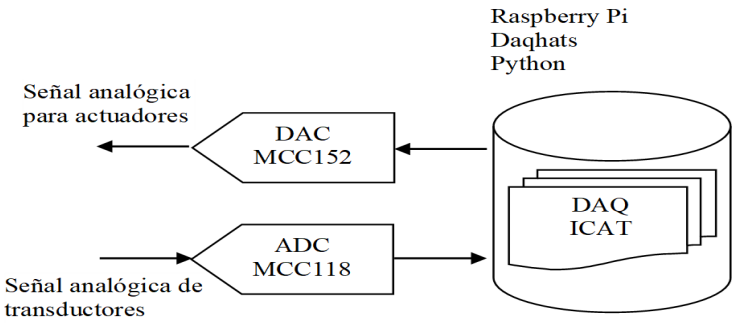
2.4 APLICACIÓN PARA LA ADQUISICIÓN Y GENERACIÓN DE SEÑALES

Basado en la configuración de *hardware* y la metodología de *software* descritas en las secciones anteriores, se desarrolló un *software* específico para implementar un sistema DAQ sobre la plataforma Raspberry Pi, como el que se muestra en la Figura 8.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 134 - 147

Figura 8

Sistema DAQ utilizando Raspberry Pi y HATS DAC y ADC



El instrumento de la Figura 8 se puede utilizar como punto de partida simple para elaborar dispositivos más complejos en donde se tengan requerimientos específicos de una aplicación particular.

El esquema de la Figura 8 muestra una Raspberry Pi configurada con la librería de paquetes daqhats y el entorno virtual de programación Python, ejecutando código de diseño específico llamado DAQ ICAT para la generación y adquisición de señales eléctricas mediante HATS DAC y ADC. Las señales analógicas del mundo real pueden ser aplicadas a diversos dispositivos actuadores o digitalizadas provenientes de diversos dispositivos transductores.

2.4.1 PROGRAMA DAQ ICAT

Para plantear el programa DAQ ICAT como un diagrama de flujo, se debe tener en cuenta que un programa que utiliza la interfaz gráfica kinter es un manejador de eventos que sólo refleja los cambios en la pantalla cuando ocurre un evento, ya sea generado por el usuario al interactuar con un control o un evento programado para generar interrupciones. En este sentido, el programa desarrollado se puede esquematizar en un diagrama de flujo como una serie de sentencias que atienden los diversos eventos. En el presente caso, los tres eventos principales de la pantalla de la Figura 9 son la atención a los botones *Inicio CDA*, *Inicio CAD* y *Fin*. De forma adicional, se programó un evento temporizador que se encarga de actualizar la muestra de salida en el generador de la onda senoidal. A estos cuatro eventos, hay que añadir el evento por omisión que inicia el programa.

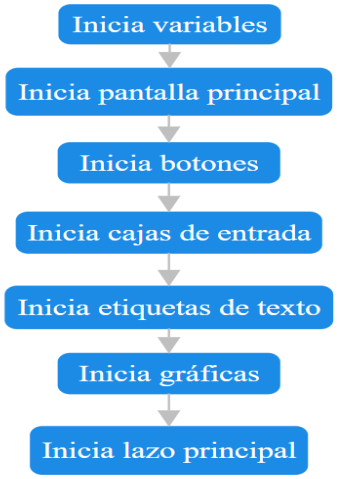
La resolución del temporizador en Python es de 1 ms y, debido a esto, la tasa a la cual se pueden escribir salidas analógicas en la tarjeta HAT CDA MCC152 es de 1 ms. Comparando esta resolución temporal, 1 ms = 1000 µs, con la tasa máxima de actualización en la MCC152 de 12 µs, se podrían generar en teoría hasta 1000 / 12 = 83 muestras cada milisegundo. El programa entonces se puede describir a partir de las 5 siguientes funciones que atienden los eventos asociados:

1. Atención al evento generado por iniciar el programa, *init*. La función *init* se encarga de iniciar la apariencia de la pantalla principal, como se muestra en la Figura 9.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 135 - 147

Figura 9

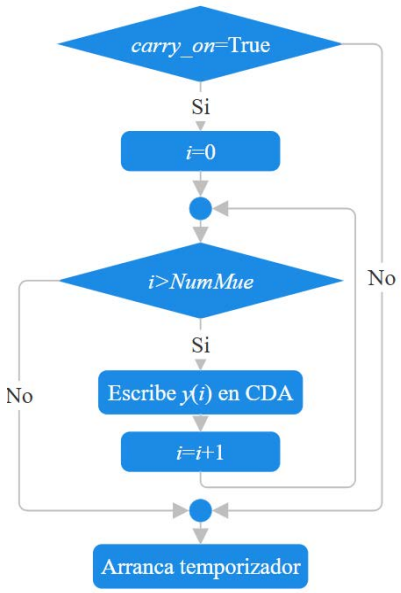
Diagrama de flujo de la función init



2. Atención al evento generado por el temporizador, *update*. La función *update* escribe el número de muestras de un periodo de la señal senoidal, almacenado y precalculado en *y*(*i*), en el DAC de la tarjeta HAT MCC152, *NumMue*, siempre y cuando la bandera *carry_on* sea verdadera y vuelve a arrancar el temporizador de 1 ms, como se muestra en el diagrama de flujo de la Figura 10.

Figura 10

Diagrama de flujo de la función update

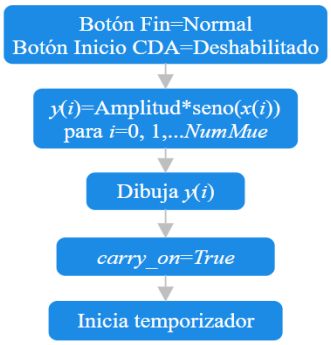


https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 136 - 147

3. Atención al evento generado por presionar el botón *Inicio CDA*, *startCDA*. La función *startCDA* actualiza el estado de los botones, calcula las muestras de la onda senoidal con la amplitud y la longitud especificadas por el usuario en *y*(*i*), dibuja el gráfico de la onda senoidal, arranca el temporizador y habilita los eventos del temporizador al establecer la bandera *carry_on* = *True* como se muestra en la Figura 11.

Figura 11

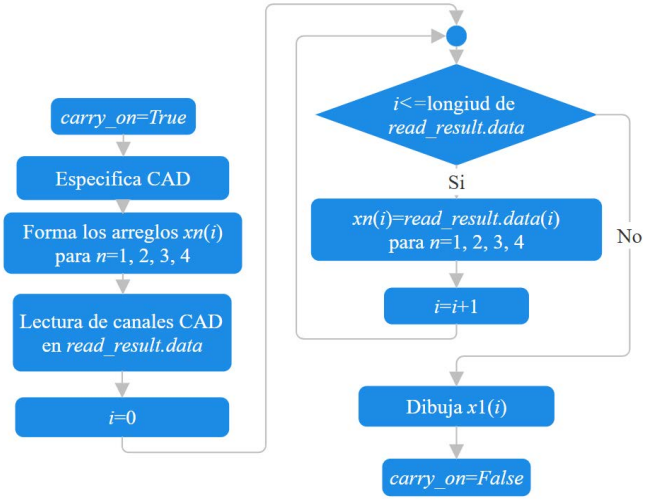
Diagrama de flujo de la función startCDA



4. Atención al evento generado por presionar el botón *Inicio CAD*, *startCAD*. La función *startCAD* habilita el temporizador con la variable *carry_on* = *True*, especifica los parámetros de conversión ADC del HAT MCC118 con el número y tasa de muestras especificadas por el usuario, prepara los arreglos para recibir cuatro canales de conversión ADC del tamaño especificado en el número de muestras, lee los cuatro canales ADC en *xn*(*i*), obtiene el resultado de la conversión en la variable *read_result.data*, dibuja el primer canal analógico y deshabilita el temporizador, como se muestra en la Figura 12.

Figura 12

Diagrama de flujo de la función startCAD

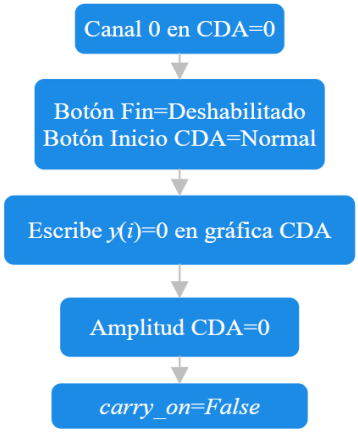


https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 137 - 147

5. Atención al evento generado por presionar el botón *Fin*, *stop*. La función *stop* escribe cero volts en el canal cero del HAT MCC152, actualiza el estado de los botones, dibuja la gráfica ADC con ceros, borra el contenido de la caja de entrada para la amplitud ADC y deshabilita el evento del temporizador con la variable *carry_on = False*, como se muestra en la Figura 13.

Figura 13

Diagrama de flujo de la función stop



En el Anexo B, se muestra el código fuente Python que implementa lo descrito en esta sección.

2.4.2 APLICACIÓN DAQ ICAT

El programa desarrollado, denominado DAQ ICAT, fue implementado en lenguaje Python y estructurado en las siguientes fases (Pidora, 2024):

1. Apariencia gráfica para el usuario (GUI). Se utiliza la librería estándar de Python Tkinter (Nikhil, 2024) para la creación de la interfaz gráfica que contempla botones y etiquetas de entrada.
2. Interfaz con las tarjetas DAQ MCC152 y MCC118. Se utiliza la librería daqhat (GitHub daqhats, 2024) que permite acceder a las funciones de escritura y lectura para la generación de señales analógicas y la adquisición de señales analógicas.
3. Graficación de la información. Se utiliza la librería matplotlib (Matplotlib Development Team. (s.f.).) para el despliegue de gráficos 2D que representen los datos en los ejes coordenados.

El concepto para desarrollar la pantalla gráfica se pensó que contuviera dos secciones:

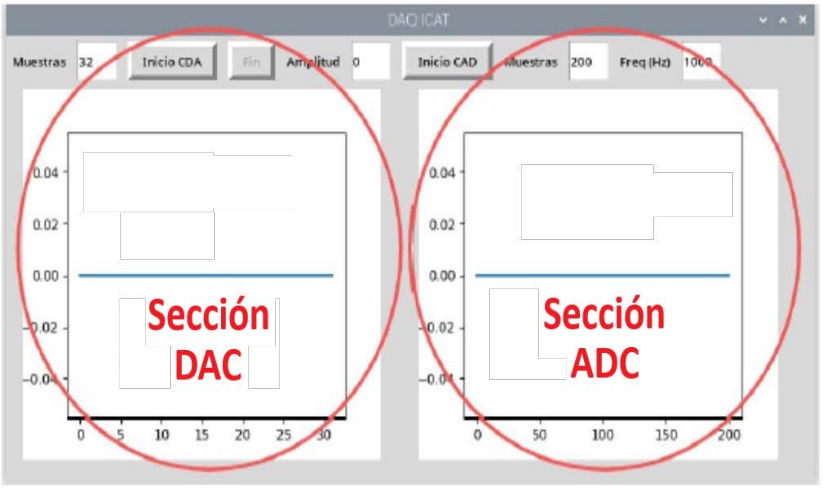
1. Generación de una señal senoidal a la máxima tasa de muestreo fija con amplitud y número de muestras variables.
2. Adquisición de cualquier señal con excursión analógica entre 0 y 5V con frecuencia de muestreo y número de muestras variables.

El concepto de la pantalla se muestra gráficamente en la Figura 14.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 138 - 147

Figura 14

Concepto para desarrollar el programa de aplicación



Cada sección contiene un área para el despliegue de datos en una gráfica de dos ejes coordenados, magnitud contra tiempo y controles para especificar los parámetros en los procesos de conversión.

3. RESULTADOS

Los resultados obtenidos corresponden a la instrumentación con base en la generación de diferentes números de muestras y a la adquisición de señales analógicas de diferentes frecuencias usando las tarjetas Raspberry Pi, HAT CDA MCC152 y MCC118. A continuación, se presentan los detalles de los resultados.

3.1 GENERACIÓN DE SEÑALES ANALÓGICAS

Para comprobar el desempeño del programa DAQ ICAT con respecto a la generación de señales, se especificaron 2 tipos de salidas con número de muestras variables y amplitud constante de 1V. La salida por omisión es del tipo senoidal y el periodo de muestreo es el mínimo que se puede obtener en Python, es decir, 1ms. El programa entonces genera cada 1 ms el número de muestras deseado por el usuario.

La Figura 15 y la Figura 16 muestran las señales analógicas especificadas para generar 8 y 128 muestras de un ciclo de señal senoidal cada 1 ms, tanto en la pantalla del programa DAQ ICAT como en la captura del osciloscopio que se utilizó para comprobar los resultados.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 139 - 147

Figura 15

Generación de señal senoidal de 8 muestras por ciclo cada 1 ms

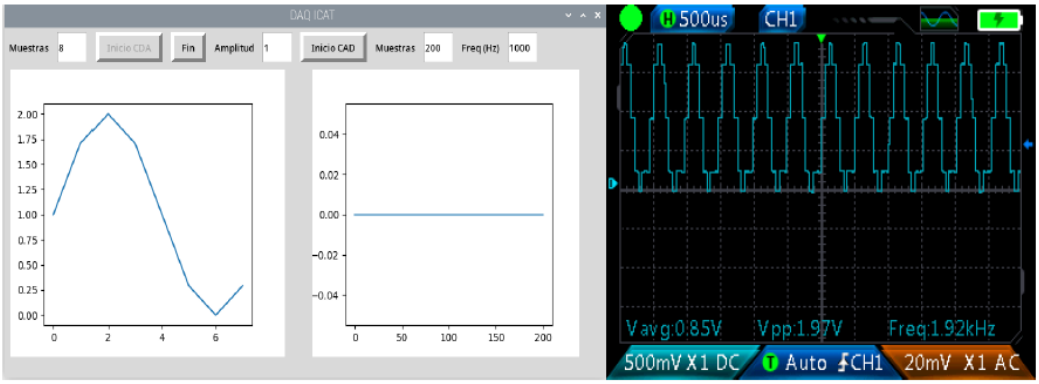
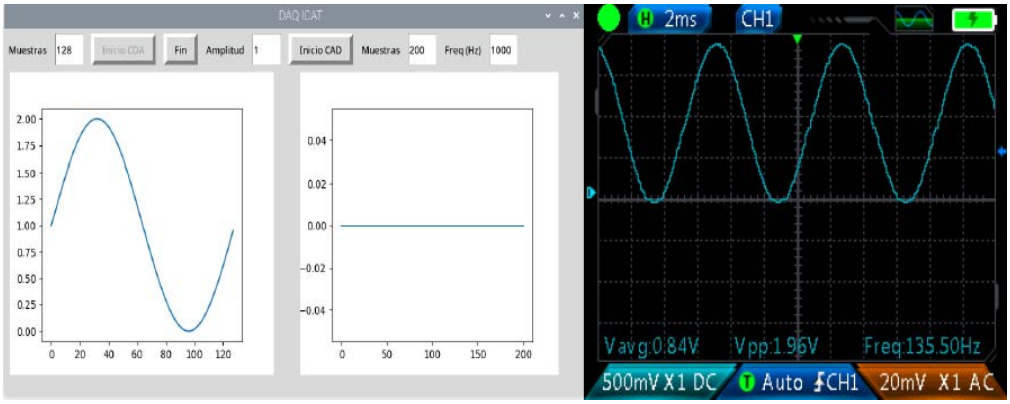


Figura 16

Generación de señal senoidal de 128 muestras por ciclo cada 1 ms



Se puede observar que, conforme aumenta el número de muestras, se incrementa la calidad de la forma de onda. Por otra parte, cuando esto sucede, disminuye la frecuencia de la forma de onda senoidal. Estos dos fenómenos deben ser tomados en cuenta en el momento de diseñar un sistema DAQ con aplicación específica.

La Tabla 1 muestra la frecuencia en la forma de onda senoidal que se puede obtener con diferente número de muestras por periodo.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 140 - 147

Tabla 1

Frecuencia de la señal senoidal para diferente número de muestras

Número de muestras	Frecuencia de la señal senoidal (Hz)
8	1920
16	900
32	483
64	242
128	135

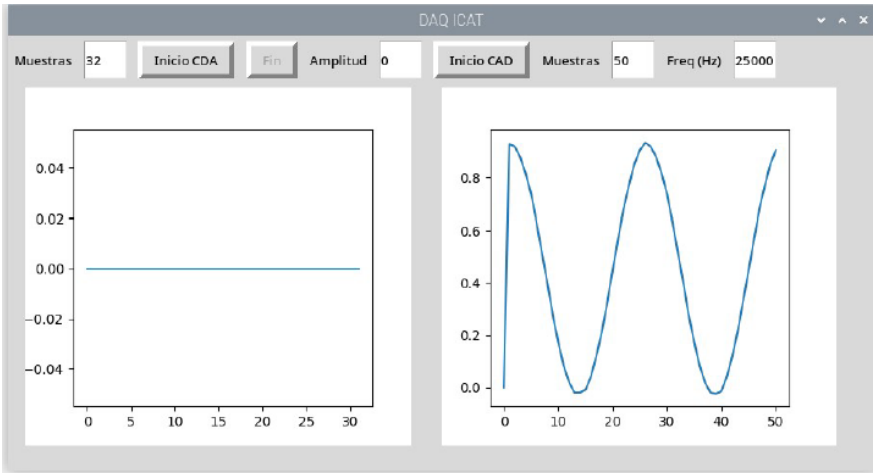
3.2 ADQUISICIÓN DE SEÑALES ANALÓGICAS

Para comprobar la sección de adquisición de señales analógicas del programa DAQ ICAT, se generaron diversas formas de onda en un generador de funciones externo y se capturaron los datos con el programa desarrollado.

La Figura 17 y la Figura 18 muestran la pantalla del generador de funciones utilizado en su modo de osciloscopio para obtener los oscilogramas de una señal senoidal y cuadrada.

Figura 17

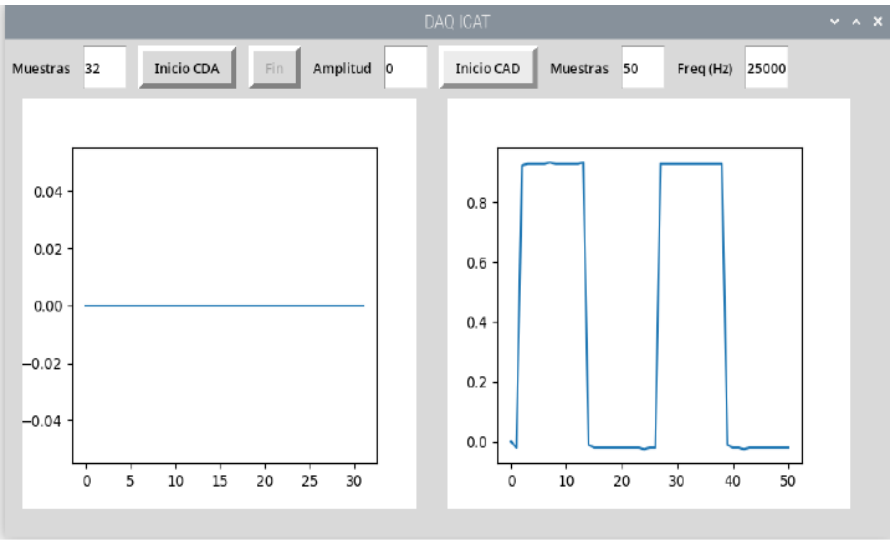
Señal senoidal de 1 kHz



https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 141 - 147

Figura 18

Señal cuadrada de 1 kHz



4. CONCLUSIONES

El desarrollo del trabajo permitió demostrar que las plataformas que trabajan con tarjetas de propósito específico, como lo son las Raspberry Pi, constituyen una alternativa viable, eficiente y de bajo costo frente a sistemas tradicionales basados en arquitecturas propietarias. La implementación de un sistema de adquisición y generación de señales (DAQ) sobre esta plataforma mostró una posible integración de soluciones de instrumentación electrónica con base en *software,* adecuadas para aplicaciones académicas, industriales y de investigación.

Entre las principales aportaciones, está la integración de módulos HAT especializados (MCC152 y MCC118) basados en la plataforma Raspberry Pi con un entorno de programación en Python, lo cual permitió no sólo la adquisición de señales analógicas, sino también su procesamiento y visualización en tiempo real mediante una interfaz gráfica. Gracias a esta interfaz, podemos validar la versatilidad de la plataforma y su capacidad de adaptación a distintos requerimientos de instrumentación de medición y control. Además, se evidenció que el uso de *software* libre y bibliotecas especializadas reduce significativamente los costos de implementación y facilita el desarrollo de soluciones personalizadas, promoviendo la independencia tecnológica y el acceso a herramientas avanzadas de instrumentación.

Respecto a los resultados obtenidos, se comprobó que la máxima tasa de muestreo en la tarjeta MCC118 es de 25,000 muestras sobre segundo, a diferencia del dato que proporciona el fabricante de 100,000 muestras sobre segundo. Por lo anterior, se presenta un desempeño adecuado en la generación de señales, donde la calidad de la forma de onda depende directamente del número de muestras utilizadas, así como en la adquisición de señales externas, logrando capturar correctamente diferentes tipos de ondas. Estos resultados validan la funcionalidad del sistema propuesto y su aplicación en entornos reales. Durante el desarrollo del *software* DAQ en un instituto de investigación, se comprobó que las soluciones

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 142 - 147

de Adafruit constituyen una alternativa a plataformas propietarias. Asimismo, se puede afirmar que las aplicaciones para la Raspberry Pi ofrecen un entorno de confiabilidad y variedad.

El sistema DAQ desarrollado puede ser de utilidad en cursos introductorios al procesamiento digital de señales, en donde se exponen conceptos como el teorema de muestreo, según los conceptos de Nyquist, o la introducción a las señales analógicas y digitales, de gran utilidad en los cursos de sistemas de comunicaciones electrónicos.

Se ha observado que las herramientas utilizadas en el desarrollo de este proyecto también son de utilidad en los cursos de programación Python y de sistemas embebidos, ya que los alumnos pueden experimentar con conceptos como la configuración del ambiente de programación y desarrollo de aplicaciones seguras que no afecten el desempeño de los sistemas operativos. Un ejemplo de esto es la configuración de ambientes virtuales como el expuesto en este reporte técnico.

Una limitación experimentada en el desarrollo de este proyecto fue la tasa de muestreo en el temporizador disponible en el ambiente de programación Python operando en el sistema operativo Raspberry Pi. En este caso, la resolución del temporizador utilizado es de 1ms, que resulta ser el estándar en la mayoría de las aplicaciones. No obstante, en otros ambientes de programación, como por ejemplo Visual C++ programando en ambiente Windows, es posible operar temporizadores de alta resolución más apropiados para aplicaciones más demandantes de recursos, como pueden ser los videojuegos. En este sentido, un trabajo a futuro podría consistir en explorar la posibilidad de utilizar o desarrollar temporizadores de alta resolución para Raspberry Pi.

Así pues, el trabajo no solo cumple con los objetivos planteados, sino que también abre la puerta a futuras líneas de investigación y desarrollo en el área de instrumentación electrónica basada en sistemas abiertos, ya que el *software* descrito en el presente reporte técnico sirve como base para elaborar aplicaciones concretas de procesamiento y control digital de señales. El sistema desarrollado puede considerarse como una plataforma escalable, susceptible de ampliación hacia aplicaciones más complejas, tales como sistemas de monitoreo remoto, procesamiento digital de señales en tiempo real o integración con tecnologías IoT. No se alcanzó el mínimo tiempo de escritura de 12 µs especificado por el fabricante de la MCC152, debido a que el tiempo de escritura en lenguaje Python es superior; no obstante las resoluciones y las excursiones analógicas son las esperadas a las que proporciona el fabricante en sus hojas de datos.

Declaración de contribución de autoría

Benjamín Valera Orozco: Instrumentación electrónica; diseño e implementación de proyecto; programación de la interfaz.

Gerardo Antonio Ruíz Botello: Metrología y gestión de la calidad; coordinador de diseño y desarrollo. Coordinación y planificación.

Karen Lucero Roldán Serrato: Cómputo aplicado; integración y creación de metodología. Integración de las componentes del proyecto

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 **Vol. 4, Núm. 2.** abril-junio 2026, págs. 143 - 147

REFERENCIAS

AG Electrónica SAPI de CV. (2024, 13 de mayo). *HAT AD de alta precisión para Raspberry Pi: ADC ADS1263 de 10 canales y 32 bits (SKU18983)* [manual técnico]. https://agelectronica.lat/pdfs/textos/S/SKU18983.PDF

Arif, R., Wijaya, S. K., Prawito, & Gani, H. S. (2018, 01-03 de mayo). *Design of EEG data acquisition system based on Raspberry Pi 3 for acute ischemic stroke identification* [ponencia]. 2018 International Conference on Signals and Systems (ICSigSys), Indonesia. https://doi.org/10.1109/ICSIGSYS.2018.8372771

Bengtsson, L. (2024). ADCs and Sampling [capítulo]. En *Electrical Measurement Techniques* (pp. 229-265). Springer. https://doi.org/10.1007/978-981-99-8187-8_11

bluemoon16. (2023, 24 de mayo). *Data Acquisition System with Raspberry Pi* [publicación en foro en línea]. Raspberry Pi Forums. https://forums.raspberrypi.com/viewtopic.php?t=351826

Cassel Barbosa, G. H., Varanis, M., Delgado, K. M. S., & Oliveira, C. de. (2020). An acquisition system framework for mechanical measurements with Python, Raspberry-Pi and MEMS sensors. *Revista Brasileira de Ensino de Física*, *42*, e20200167. https://doi.org/10.1590/1806-9126-RBEF-2020-0167

Chen, L. X., & Liu, Y. T. (2025). Implementation of signal acquisition system (SAS) for large voltage waveform using low-cost Raspberry Pi Pico. *Preprints*. https://doi.org/10.20944/preprints202505.1574.v1

Ellison Mathe, S., Kondaveeti, H. K., Vappangi, S., Vanambathina, S. D., & Kumaravelu, N. K. (2024). A comprehensive review on applications of Raspberry Pi. *Computer Science Review, 52*, 100636. https://doi.org/10.1016/j.cosrev.2024.100636

Gunawan, T. S., Rahman, S. N., Kartiwi, M., & Ihsanto, E. (2020, 23-24 de octubre). *Development of Very Low Frequency Data Acquisition System using Raspberry Pi* [ponencia]. 2020 8th International Conference on Cyber and IT Service Management (CITSM), Indonesia. https://doi.org/10.1109/CITSM50537.2020.9268882

Jaryd. (2023, 7 de diciembre). *Using virtual environments in Thonny on a Raspberry Pi*. Core Electronics. https://core-electronics.com.au/guides/using-virtual-environments-in-thonny-on-a-raspberry-pi/

Jolles, J. W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. *Methods in Ecology and Evolution, 12*(9), 1562–1579. https://doi.org/10.1111/2041-210X.13652

Kondo, K., Tanno, K., Tamura, H., & Nakatake, S. (2018). Low voltage CMOS current mode reference circuit without operational amplifiers. *IEICE Transactions on Fundamentals, 101*(5), 748–754. https://doi.org/10.1587/transfun.E101.A.748

Llamas, L. (2025). *Raspberry Pi pinout diagram*. https://www.luisllamas.es/en/raspberry-pi-pinout/

Magalhães, R. (2024, 25 de junio). *Cómo leer señales analógicas en Raspberry Pi usando ADS1015/ADS1115*. Compraco. https://compraco.com.br/es/blogs/tecnologia-e-desenvolvimento/como-ler-sinais-analogicos-no-raspberry-pi-usando-ads1015-ads1115

Martínez-Quintero, J. C., Estupiñán-Cuesta, E. P., & Rodríguez-Ortega, V. D. (2019). Raspberry Pi 3 RF signal generation system. *Visión Electrónica, 13*(2). https://doi.org/10.14483/22484728.15160

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 144 - 147

Matplotlib Development Team. (s.f.). *Matplotlib 3.9.2 documentation*. https://matplotlib.org/stable/

Measurement Computing Corporation. (2019). *MCC 152 voltage output and DIO DAQ HAT for Raspberry Pi* [manual técnico]. https://files.digilent.com/datasheets/DS-MCC-152.pdf

Measurement Computing Corporation. (2020). *MCC 118 voltage measurement DAQ HAT for Raspberry Pi: Datasheet (DS-MCC-118)* [manual técnico]. https://files.digilent.com/datasheets/DS-MCC-118.pdf

Measurement Computing Corporation. (2024). *daqhats: MCC DAQ HAT Library for Raspberry Pi* (v1.5.0.0) [repositorio de software]. https://github.com/mccdaq/daqhats

Medina de Andrés, D. (2017). *Diseño e implementación de un sistema de adquisición y procesado digital de señales biomédicas* [trabajo de fin de grado]. Universidad Politécnica de Madrid. https://oa.upm.es/52911/1/TFG_DAVID_MEDINA_DE_ANDRES.pdf

Meena, T., Sharma, V., Jhakar, S., & Sharma, S. K. (2019). Raspberry Pi based electromyography signal acquisition and processing system. *Journal of Emerging Technologies and Innovative Research, 6*(4), 69–75. https://www.jetir.org/papers/JETIR1904E09.pdf

Nikhil (2024). *Python Tkinter tutorial*. https://www.geeksforgeeks.org/python-tkinter-tutorial/

Pidora. (2024, 4 de marzo). *Transforma tu Raspberry Pi en una potente estación de procesamiento de señales*. https://pidora.ca/transform-your-raspberry-pi-into-a-powerful-signal-processing-workstation/

Razavi, B. (2021). *Fundamentals of microelectronics*. Wiley.

Rodríguez Corbo, F. A., Hernández González, A., & Ramírez Beltrán, J. (2018). Adquisición de datos analógicos con alta precisión usando una BeagleBone Black. *RIELAC: Revista Electrónica de Ingeniería, Electrónica, Automática y Comunicaciones, 39*(3), 68–76. https://dialnet.unirioja.es/descarga/articulo/6676696.pdf

Sreejith, A. G., Mathew, J., Sarpotdar, M., Mohan, R., Nayak, A., Safonova, M., & Murthy, J. (2014, noviembre). A Raspberry Pi-Based Attitude Sensor. *Journal of Astronomical Instrumentation, 3*(2). https://doi.org/10.1142/S2251171714400066

XOREN Ingeniería. (2023. 16 de noviembre). Consejo técnico: adquisición de datos en Raspberry Pi con Universal Library para Linux. https://xoreningenieria.com/2023/11/16/consejo-tecnico-adquisicion-de-datos-en-raspberry-pi-con-universal-library-para-linux/

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 145 - 147

ANEXO A. GLOSARIO

DAQ—*Data Acquisition System* (Sistema de Adquisición de Datos)

Es uno de los procesos más importantes del sistema y está compuesto por *hardware* y *software* que permite medir variables físicas, acondicionar señales cuyas fuentes son de sensores y convertirlas a un formato digital para su procesamiento. En el tema de instrumentación, es el medio por el cual en un sistema IoT registra en tiempo real un fenómeno físico (Razavi, 2021; Bengtsson, 2024).

DAC — *Digital-to-Analog Converter* (Convertidor Digital–Analógico)

Un DAC convierte información digital (generalmente en formato binario) en una señal analógica continua, como voltaje o corriente. En instrumentación, se utiliza para generar señales de prueba, controlar actuadores y crear referencias analógicas. Este principio es fundamental para sistemas IoT, ya que permite que un dispositivo produzca salidas analógicas para control o simulación (Kondo *et al.*, 2018).

ADC — *Analog-to-Digital Converter* (Convertidor Analógico–Digital)

El convertidor ADC transforma una señal analógica en un valor digital cuantificado que puede ser manejado por un microcontrolador o procesador. En instrumentación, se emplea para adquirir señales de sensores; en IoT, permite que los nodos inteligentes midan variables del entorno (temperatura, luz, presión, etc.) (Razavi, 2021; Bengtsson, 2024).

HAT — *Hardware Attached on Top* (Tarjeta de Expansión para Raspberry Pi)

Un HAT es una tarjeta de expansión estandarizada que se monta directamente sobre la Raspberry Pi para agregar funcionalidades adicionales, como ADC/DAC de precisión, interfaces industriales o módulos de comunicación. En instrumentación, permite convertir la Raspberry Pi en un módulo DAQ; en IoT, permite ampliar el sistema con conectividad LoRa, NB-IoT, GPS u otros (Jolles, 2021).

ANEXO B. CÓDIGO FUENTE

```
# GUI
from tkinter import *
# Vectores
import numpy as np
# Graficas
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
# DAQ DAC Hat
from sys import version_info
from daqhats import mcc152, OptionFlags, HatIDs, HatError
# DAQ ADC Hat
from daqhats import mcc118, OptionFlags, HatIDs, HatError
class mainwindow(Tk):
def __init__(self, title):
# variables timer
self.carry_on = True
# variables DAC
```

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 146 - 147

```
self.i=0
self.optionsCDA=OptionFlags.DEFAULT
self.channelCDA=0
self.hatCDA=mcc152(0)
self.length = np.pi * 2
self.x=np.arange(0, self.length, self.length / 32)
self.y = (np.sin(self.x)*0)+0
self.figCDA = Figure(figsize = (4, 3.7), dpi = 100)
self.plotCDA = self.figCDA.add_subplot(111)
# variables ADC
self.optionsCAD=OptionFlags.DEFAULT
self.hatCAD=mcc118(1)
self.y1 = np.arange(201)
self.x1 = self.y1*0.0
self.x2 = self.y1*0.0
self.x3 = self.y1*0.0
self.x4 = self.y1*0.0
self.figCAD = Figure(figsize = (4, 3.7), dpi = 100)
self.plotCAD = self.figCAD.add_subplot(111)
# GUI
self.master = Tk()
self.master.title(title)
self.master.geometry('900x450')
self.window = Frame(self.master, bd=0, relief=FLAT)
self.window.grid( column = 0, row = 0)
self.NumSen = Label(self.window, text = 'Muestras')
self.NumSen.grid(row=0, column=0, sticky='NSWE', padx=5, pady=5)
self.NumMue = Entry(self.window, width=5, text = '0')
self.NumMue.grid(row=0, column=1, sticky='NSWE', padx=5, pady=5)
self.NumMue.insert(0, "32")
self.btnCDA = Button(self.window, text = 'Inicio CDA', bd = '5', command =
self.startCDA)
self.btnCDA.grid(row=0, column=2, sticky='NSWE', padx=5, pady=5)
self.btnFinCDA = Button(self.window, text = 'Fin', bd = '5', command =
self.stop)
self.btnFinCDA.grid(row=0, column=3, sticky='NSWE', padx=5, pady=5)
self.btnFinCDA.config(state=DISABLED)
self.Amplitud = Label(self.window, text = 'Amplitud')
self.Amplitud.grid(row=0, column=4, sticky='NSWE', padx=5, pady=5)
Self.Ampl = Entry(self.window, width=5, text = '1')
self.Ampl.grid(row=0, column=5, sticky='NSWE', padx=5, pady=5)
self.Ampl.insert(0, "0")
self.btnCAD = Button(self.window, text = 'Inicio CAD', bd = '5', command =
self.startCAD)
self.btnCAD.grid(row=0, column=6, sticky='NSWE', padx=5, pady=5)
self.Muestras = Label(self.window, text = 'Muestras')
self.Muestras.grid(row=0, column=7, sticky='NSWE', padx=5, pady=5)
self.Num = Entry(self.window, width=5, text = '2')
self.Num.grid(row=0, column=8, sticky='NSWE', padx=5, pady=5)
self.Num.insert(0, "200")
self.Frecuencia = Label(self.window, text = 'Freq (Hz)')
self.Frecuencia.grid(row=0, column=9, sticky='NSWE', padx=5, pady=5)
self.Freq = Entry(self.window, width=5, text = '3')
self.Freq.grid(row=0, column=10, sticky='NSWE', padx=5, pady=5)
self.Freq.insert(0, "1000")
self.plotCDA.plot(self.y)
canvasCDA = FigureCanvasTkAgg(self.figCDA,master=self.master)
canvasCDA.draw()
canvasCDA.get_tk_widget().place(relx=0.02, rely=0.12)
self.plotCAD.plot(self.x1)
canvasCAD = FigureCanvasTkAgg(self.figCAD,master=self.master)
canvasCAD.draw()
canvasCAD.get_tk_widget().place(relx=0.51, rely=0.12)
self.master.mainloop()
def update(self):
if self.carry_on:
for j in range(int(self.NumMue.get())):
value = self.y[j]
self.hatCDA.a_out_write(channel=self.channelCDA, value=value,
options=self.optionsCDA)
self.master.after(1, self.update)
def startCDA(self):
self.btnFinCDA.config(state=NORMAL)
self.btnCDA.config(state=DISABLED)
```

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.146 Vol. 4, Núm. 2. abril-junio 2026, págs. 147 - 147

```
self.x=np.arange(0, self.length, self.length / int(self.NumMue.get()))
Amplitud=float(self.Ampl.get())
self.y = (np.sin(self.x)*Amplitud)+Amplitud
print(self.y)
self.plotCDA.clear()
self.plotCDA.plot(self.y)
canvasCDA = FigureCanvasTkAgg(self.figCDA,master=self.master)
canvasCDA.draw()
canvasCDA.get_tk_widget().place(relx=0.01, rely=0.12)
self.carry_on = True
self.master.after(1, self.update)
self.btnFinCDA.config(state=NORMAL)
def startCAD(self):
self.carry_on = True
channel_mask=15 #4 canales
samples_per_channel=int(self.Num.get())
scan_rate=int(self.Freq.get())
num_channels = 4
self.hatCAD.a_in_scan_start(channel_mask, samples_per_channel, scan_rate,
self.optionsCAD)
self.y1 = np.arange(samples_per_channel+1)
self.x1 = self.y1*0.0
self.x2 = self.y1*0.0
self.x3 = self.y1*0.0
self.x4 = self.y1*0.0
total_samples_read = 0
read_request_size = samples_per_channel
timeout = 5.0
while total_samples_read < samples_per_channel:
read_result = self.hatCAD.a_in_scan_read(read_request_size, timeout)
# Check for an overrun error
if read_result.hardware_overrun:
print('\n\nHardware overrun\n')
break
elif read_result.buffer_overrun:
print('\n\nBuffer overrun\n')
break
samples_read_per_channel = int(len(read_result.data) / num_channels)
total_samples_read += samples_read_per_channel
print(total_samples_read)
if samples_read_per_channel > 0:
i=0
ii=1
while i<=len(read_result.data)-4:
self.x1[ii]=read_result.data[i]
i=i+1
self.x2[ii]=read_result.data[i]
i=i+1
self.x3[ii]=read_result.data[i]
i=i+1
self.x4[ii]=read_result.data[i]
i=i+1
ii=ii+1
i=0
while i<samples_per_channel:
i=i+1
print(i,self.x1[i])
self.plotCAD.clear()
self.plotCAD.plot(self.x1)
canvasCAD = FigureCanvasTkAgg(self.figCAD,master=self.master)

canvasCAD.get_tk_widget().place(relx=0.5, rely=0.12)
self.master.after(1, self.update)
self.hatCAD.a_in_scan_cleanup()
self.carry_on = False
def stop(self):
self.hatCDA.a_out_write(channel=self.channelCDA, value=0,
options=self.optionsCDA)
self.btnFinCDA.config(state=DISABLED)
self.btnCDA.config(state=NORMAL)
Amplitud=0
self.y = (np.sin(self.x)*Amplitud)+Amplitud
print(self.y)
self.plotCDA.clear()
self.plotCDA.plot(self.y)
canvasCDA = FigureCanvasTkAgg(self.figCDA,master=self.master)
canvasCDA.draw()
canvasCDA.get_tk_widget().place(relx=0.01, rely=0.12)
self.Ampl.delete(0, 'end')
self.Ampl.insert(0, "0")
self.carry_on = False
if __name__ == '__main__':
mainwin = mainwindow( "DAQ ICAT" )
```