

# 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:

Licencia Creative Commons



El contenido de los textos es responsabilidad de los autores y no refleja forzosamente el punto de vista de los dictaminadores, o de los miembros del Comité Editorial, o la postura del editor y la editorial de la publicación.

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](mailto: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](mailto: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](mailto: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

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.

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<sup>2</sup>C 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

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

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.

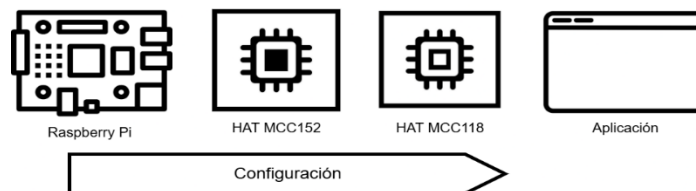
## 2.1 INTEGRACIÓN DE CIRCUITOS DE SEÑALES CON RASPBERRY PI

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.

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  $\mu$ s.

#### **Adquisición de señales**

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

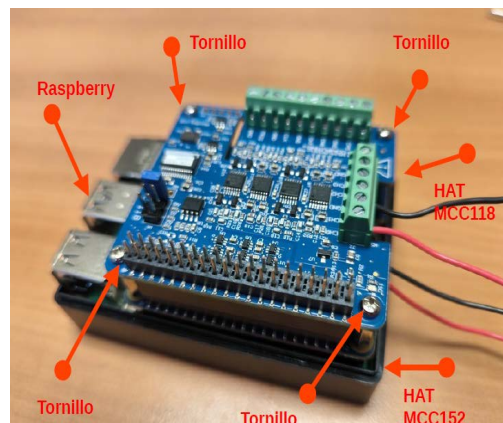
## 2.2 CONFIGURACIÓN DE HARDWARE DEL SISTEMA ELECTRÓNICO DAQ

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*



## 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

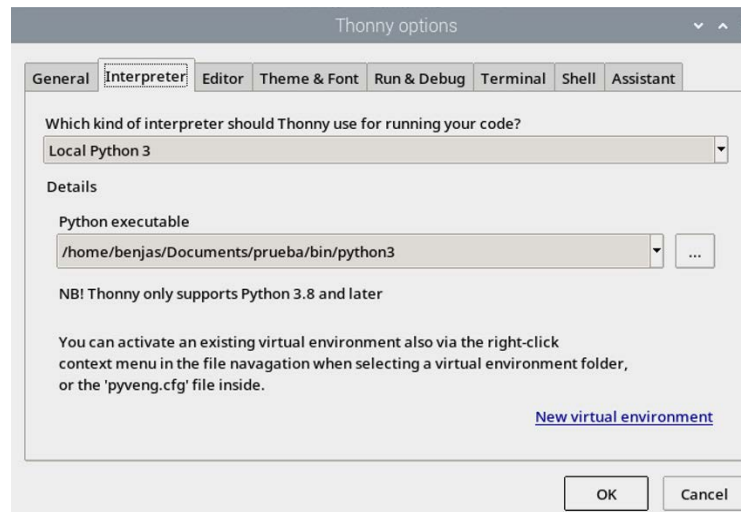
Para verificar la correcta operación de las configuraciones descritas en 2.4.1, se utiliza Thony como ambiente de programación (Jaryd, 2023) en la constitución de un proyecto de aplicación mediante el cual se pueda comprobar la correcta operación de las tarjetas HAT.

En este sentido y con fines de seguridad computacional, se crea un entorno virtual de programación que aisle 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.

### Figura 3

Diálogo "Thonny options" para crear un nuevo ambiente virtual

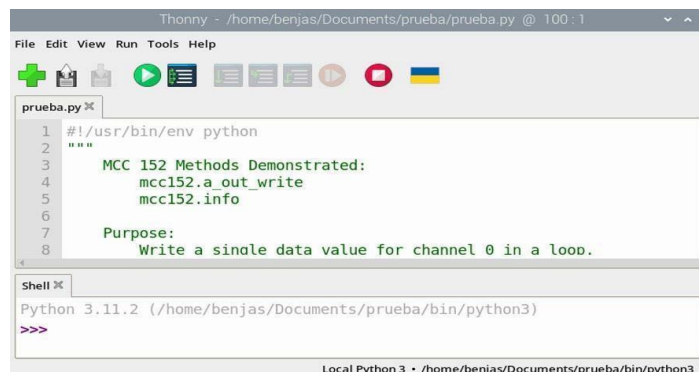


Al presionar el enlace *New virtual environment* de la Figura 3, se despliega una advertencia en donde se hace énfasis que el directorio a seleccionar debe estar vacío para el nuevo ambiente virtual. Al aceptar la advertencia, se selecciona el directorio vacío de prueba y se retorna al diálogo de la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

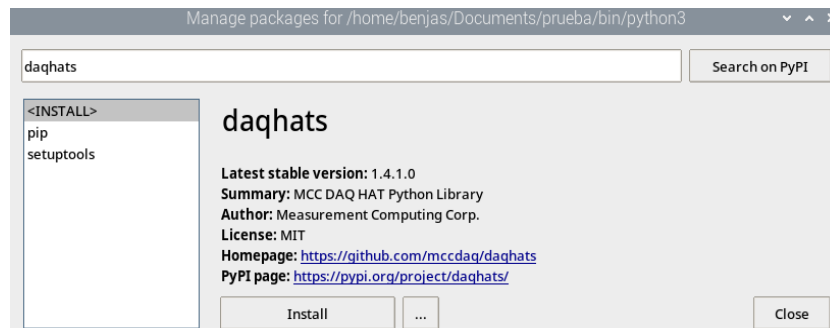
Programa de ejemplo en el archivo *prueba.py*



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
```

Línea 71 se enmascara la instrucción:

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

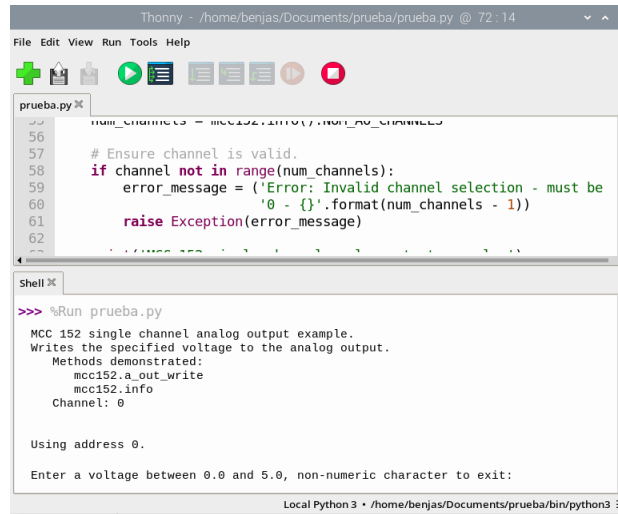
Se agrega la línea 72:

```
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.

**Figura 6**

*Programa prueba.py ejecutándose libre de errores*

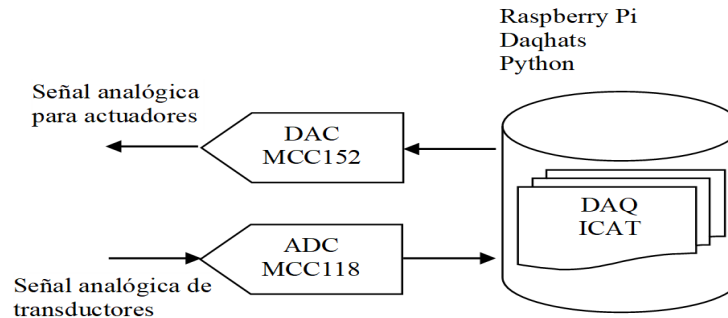


```
Thonny - /home/benjas/Documents/prueba/prueba.py @ 72 : 14
File Edit View Run Tools Help
prueba.py
56 num_channels = MCC152.A152V1.NUM_CHANNELS
57
58 # Ensure channel is valid.
59 if channel not in range(num_channels):
60     error_message = ('Error: Invalid channel selection - must be
61                     '0 - {}'.format(num_channels - 1))
62     raise Exception(error_message)
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611

```

**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

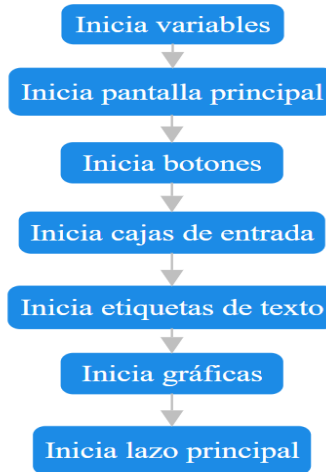
Para plantear el programa DAQ ICAT como un diagrama de flujo, se debe tener en cuenta que un programa que utiliza la interfaz gráfica kinter es un manejador de eventos que sólo refleja los cambios en la pantalla cuando ocurre un evento, ya sea generado por el usuario al interactuar con un control o un 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,  $1\text{ ms} = 1000\ \mu\text{s}$ , con la tasa máxima de actualización en la MCC152 de  $12\ \mu\text{s}$ , 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.

**Figura 9**

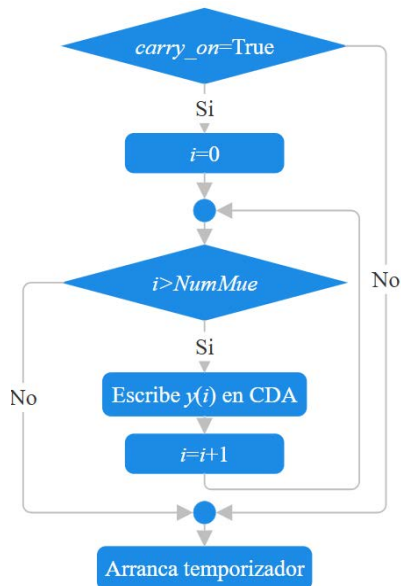
Diagrama de flujo de la función *init*



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**

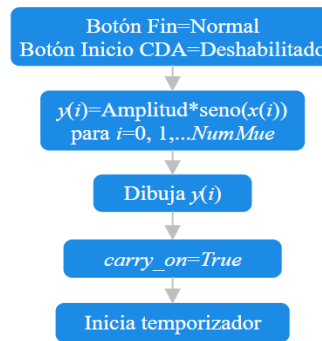
Diagrama de flujo de la función *update*



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**

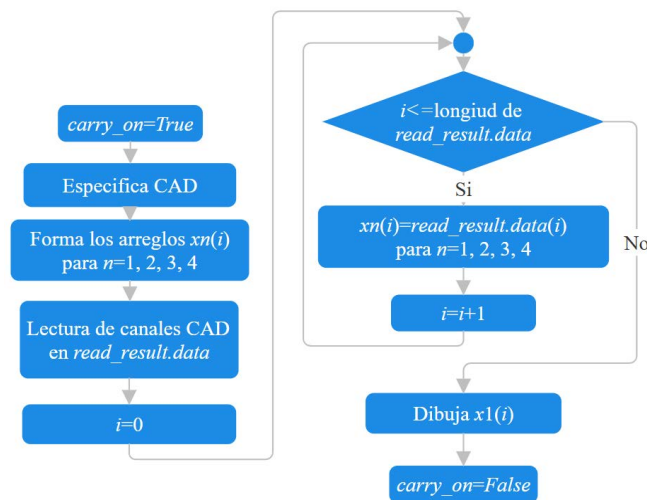
*Diagrama de flujo de la función startCDA*



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**

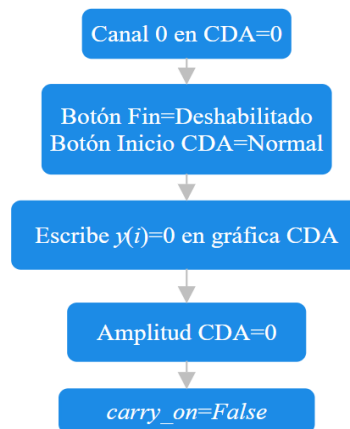
*Diagrama de flujo de la función startCAD*



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.

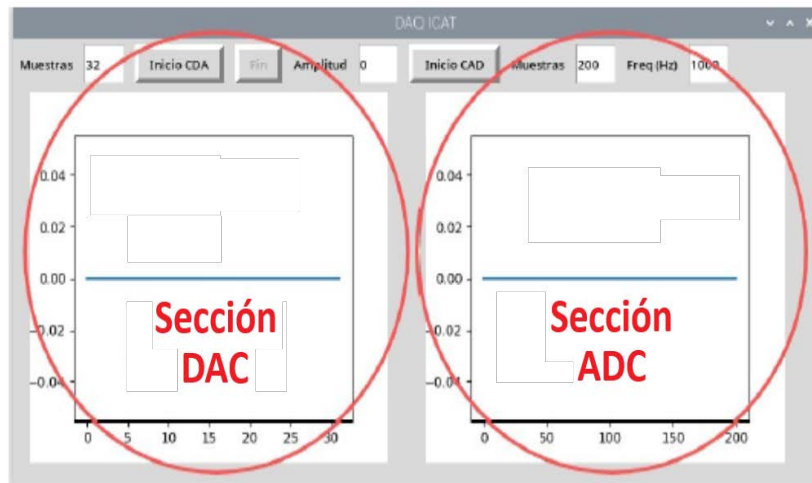
El concepto para desarrollar la pantalla gráfica se pensó que contuviera dos secciones:

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.

**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.

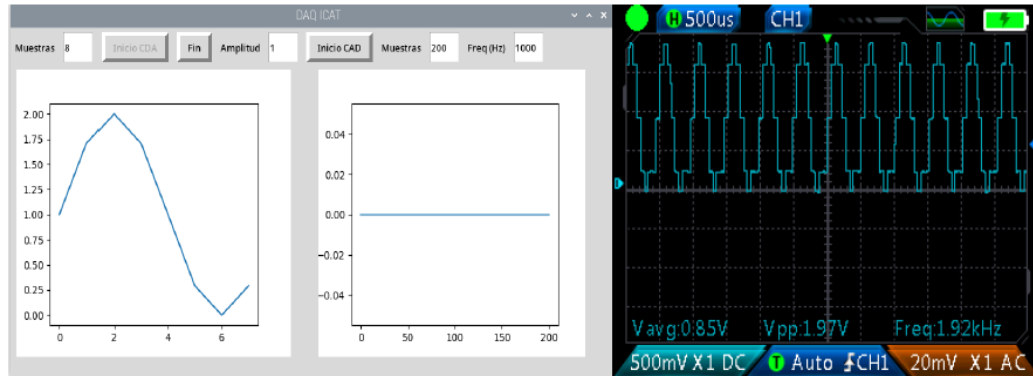
#### 3.1 GENERACIÓN DE SEÑALES ANALÓGICAS

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.

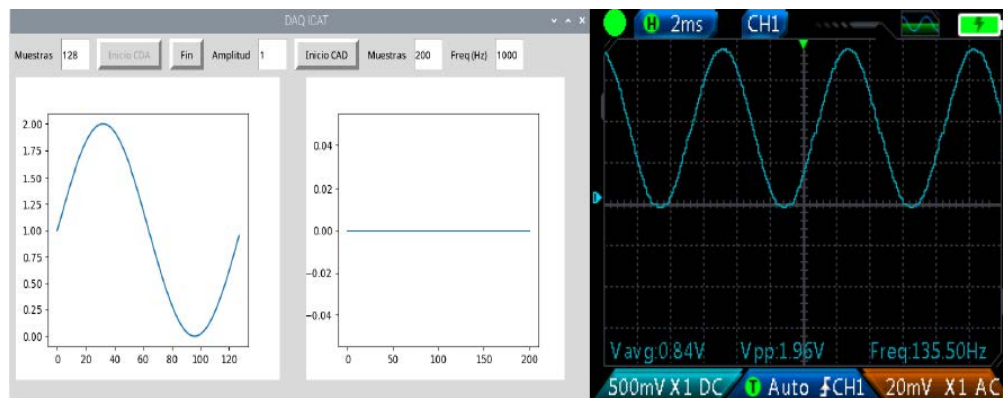
**Figura 15**

*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.

**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

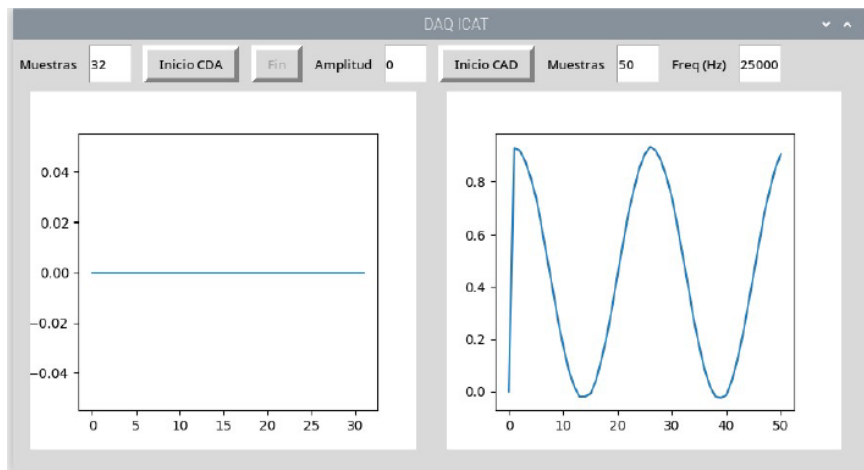
### 3.2 ADQUISICIÓN DE SEÑALES ANALÓGICAS

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.

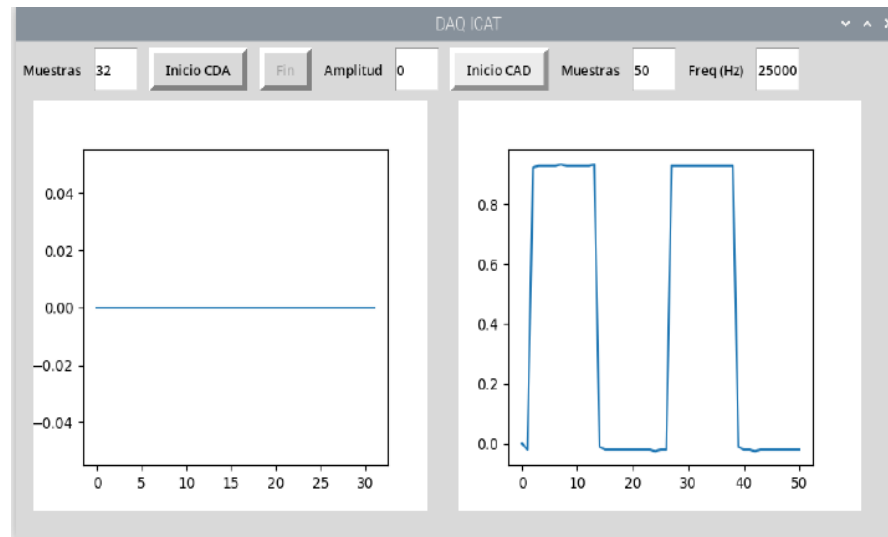
**Figura 17**

*Señal senoidal de 1 kHz*



**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

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  $\mu$ s 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

## 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](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>

- 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 procesamiento 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](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://xoreningeneria.com/2023/11/16/consejo-tecnico-adquisicion-de-datos-en-raspberry-pi-con-universal-library-para-linux/>

## 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):
import # variables timer
self.carry_on = True
# variables DAC
```

```

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)

```

```

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")

```