

CÁTEDRA DE PROYECTO FINAL / INGENIERÍA EN ELECTRÓNICA



PROYECTO FINAL DE CARRERA

DISPOSITIVO DE MONITOREO DE
VIBRACIONES PARA SALUD
ESTRUCTURAL

07/04/2022

Autor: Fernando Guerrero

Mail: fernandoguerrero662@gmail.com

Teléfono: +54 9 263 4555138

RESUMEN

El proyecto realizado corresponde al diseño y desarrollo de un dispositivo detector y medidor de vibraciones. El mismo ha sido implementado a través de una placa de desarrollo Raspberry Pi, un conversor analógico/digital de alta resolución y sensores medidores de vibraciones. El dispositivo final logrado posee la capacidad de realizar lecturas de forma continua y almacenarlas para su posterior análisis.

El objetivo final del producto desarrollado es lograr mediciones confiables de buena calidad que en un futuro serán empleadas en análisis de salud estructural.

Las grandes ventajas que traerá este equipo será la de poseer un diseño propio y ajustable, que pueda ser modificado y ampliado; implementado de forma económica y fácilmente accesible en el mercado argentino.

Este proyecto surge a partir de la necesidad del Área Dinámica Experimental del Instituto de Mecánica Estructural y Riesgo Sísmico (IMERIS), de obtener un dispositivo de medición de vibraciones que sea económico, confiable y cuyo diseño y desarrollo sea de su propiedad. Su objetivo principal es determinar respuestas a estructuras y suelos sometidos a cargas extremas y proponer métodos de evaluación y mitigación del daño estructural, y es aquí donde el dispositivo desarrollado brindará una solución.

ÍNDICE

Contenido

RESUMEN	1
1 INTRODUCCIÓN	4
1.1 OBJETIVO GENERAL	4
1.2 OBJETIVOS PARTICULARES	4
2 ANTENCEDENTES	5
2.1 ESTADO ACTUAL	5
2.2 DEFINICIÓN DEL PROBLEMA	5
3 ESPECIFICACIONES	7
3.1 CARACTERÍSTICAS ELÉCTRICAS	8
4 INGENIERÍA GENERAL	9
4.1 REQUISITOS MÍNIMOS	9
4.2 ALCANCES	10
4.3 FUNCIONAMIENTO GENERAL	11
4.4 DIAGRAMA EN BLOQUES	16
5 INGENIERÍA DE DETALLE	18
5.1 HARDWARE	18
5.1.1 Raspberry Pi 3 B+	18
5.1.2 Placa conversora AD/DA de alta precisión:	20
5.1.3 Circuito de protección:	25
5.1.4 Módulo RTC:	33
5.1.5 Esquema de conexión completo:	35
5.2 SOFTWARE	38
5.2.1 REGISTROS	40
5.2.2. COMANDOS	50
5.2.3 LIBRERÍAS E INSTRUCCIONES	55
5.2.4 PROGRAMA DE MEDICIÓN	66
6 INSTALACIÓN DEL MÓDULO RTC	96
7 INTERFAZ DE USUARIO	102
7.1 ACCESO REMOTO DENTRO DE LA RED DE ÁREA LOCAL	102
7.2 ACCESO REMOTO FUERA DE LA RED DE ÁREA LOCAL	108
7.2.1 LIBRERÍAS E INSTRUCCIONES PRINCIPALES	115

7.2.2 PROGRAMA PARA COMUNICACIÓN REMOTA DESDE TELEGRAM .	118
7.3 TRASNFERENCIA DE ARCHIVOS A SERVIDOR FTP	165
7.4 EXPERIENCIA.....	167
8 EJECUCIÓN AUTOMÁTICA.....	180
9 PRUEBAS REALIZADAS	185
10 ANEXO: Manual de instalación.....	222
• Paso 1: Conexión de los dispositivos.	222
• Paso 2: Instalación de un sistema operativo.	229
• Paso 3: Acceso al dispositivo.	230
• Paso 4: Instalación de librerías y compilador.	235
• Paso 5: Programas principales.....	237
• Paso 6: Archivos adjuntos.....	238
• Paso 7: Creación del bot de Telegram.	239
• Paso 8: Instalación de módulo RTC.	243
• Paso 9: Creación de servicios y mensajes de inicio.	248
11 ANEXO: Manual de Usuario.....	253
11.1 Conexionado.....	253
11.2 Encendido y acceso al dispositivo.....	254
11.3 Parámetros de configuración	259
11.4 Manejo de archivos	261
11.5 Chat de Telegram	264
12 HOJAS DE DATOS	276
12.1 Raspberry Pi 3 Model B+	276
12.2 Placa conversora AD-DA	280
12.3 Conversor ADS1256	284
12.4 Módulo RTC DS3231	288
12.5 Circuito de protección	291
12.5.4 Diseño PCB	298
13 BIBLIOGRAFÍA.....	300

1 INTRODUCCIÓN

1.1 OBJETIVO GENERAL

El objetivo general del proyecto es lograr la obtención de un dispositivo capaz de medir las vibraciones producidas en estructuras de interés y almacenarlas en archivos, con sus respectivas estampas de tiempo, para su posterior descarga. Por lo tanto, se debe obtener como producto final, un dispositivo de monitoreo de vibraciones y almacenamiento de muestras con acceso directo y remoto a los archivos generados.

El usuario podrá modificar ciertos parámetros de configuración, parámetros que le permitirán un control preciso del dispositivo y de la forma en que éste toma las mediciones.

Se hará uso de una placa Raspberry Pi modelo 3 B+, que es un ordenador de placa reducida de bajo costo. Dicha placa será programada a través del lenguaje C y utilizando librerías especiales, tanto para el uso del conversor analógico/digital, como también para el manejo de archivos y del tiempo.

El conversor analógico/digital elegido para realizar la toma de muestras y su posterior conversión es el conversor ADS1256. Se trata de un conversor de alta resolución de 24 bits y bajo ruido.

El sistema operativo instalado en la placa Raspberry Pi es Raspbian GNU/Linux 10 Buster, una distribución del sistema operativo GNU/Linux basado en Debian y que es por lo tanto libre para la SBC Raspberry Pi.

Estas son las herramientas principales para lograr el objetivo general, que es obtener un dispositivo confiable, de fácil utilización y que responda de forma adecuada a las situaciones y especificaciones propuestas.

1.2 OBJETIVOS PARTICULARES

1. Como objetivo particular se debe lograr el correcto funcionamiento en cada etapa desarrollada del proyecto, con el fin de obtener un producto final que cumplirá con el objetivo general planteado anteriormente.
2. Realizar un estudio detallado de los dispositivos a usar, con el objetivo de conocer su funcionamiento y sus limitaciones.
3. Realización de diversas pruebas en cada etapa y sometimiento del dispositivo a distintas situaciones probables y adversas para obtener un desarrollo completo y confiable. A través de sus resultados determinar si el dispositivo cumple los requisitos mínimos impuestos.
4. Lograr un código entendible, bien estructurado y robusto, que tenga en cuenta y solucione los problemas que puedan ocurrir al usar el dispositivo.
5. Entregar un documento bien detallado que hará de manual de usuario, donde se especificará tanto el desarrollo de todo el dispositivo, como también los resultados obtenidos en cada prueba, las limitaciones del producto, recomendaciones y precauciones.

2 ANTENCIÉDENTES

El Área Dinámica Experimental del Instituto de Mecánica Estructural y Riesgo Sísmico (IMERIS) trata temas relacionados con las acciones dinámicas y sus efectos sobre sistemas estructurales y mecánicos, en relación con el estudio de vibraciones. Tiene como objetivo principal determinar respuestas a estructuras y suelos sometidos a cargas extremas, como explosiones, impacto, sismo o viento; además proponer métodos de evaluación y mitigación del daño estructural generado por las mismas. Este objetivo se cumple mediante el desarrollo de nuevas tecnologías para la reducción de vibraciones en estructuras sometidas a acciones dinámicas.

Este instituto requiere dispositivos de medición y monitoreo para realizar mediciones y análisis que permitan cumplir sus principales objetivos.

Es en base a esto que surge el origen de este proyecto. Se requiere un dispositivo de medición de vibraciones que sea económico, confiable y un desarrollo propio del mismo y es aquí donde el dispositivo en cuestión brindará una solución apropiada.

2.1 ESTADO ACTUAL

Actualmente, el Área Dinámica Experimental posee el dispositivo USB-1608G que es un dispositivo DAQ multifunción de alta velocidad de 16 bits. Este dispositivo posee un gran nivel de tecnología y es de una excelente calidad.

Las principales desventajas de este dispositivo son su elevado precio y el monopolio en Argentina para su importación mediante un único representante. Su precio ronda, según el modelo y las especificaciones, entre US\$ 440 y US\$ 880 más el software complementario para la toma de muestras de US\$ 50. Debido a que estos dispositivos deben instalarse en las estructuras a analizar, conlleva un elevado riesgo dejar dispositivos de este valor en tales sitios, con el siempre posible riesgo de hurto o vandalismo.

2.2 DEFINICIÓN DEL PROBLEMA

La necesidad principal a cubrir es la falta de un dispositivo de adquisición de datos analógicos cuyo diseño y desarrollo sea propiedad de la institución y sea de bajo coste. Esto se debe a que las alternativas actuales del dispositivo que se ha propuesto a desarrollar son sistemas cerrados, lo que presenta la desventaja de que no pueden ser modificados a gusto según la aplicación o el uso específico que se le desee dar, referidos tanto al software como al hardware. A esto se suma que los costos de dichos dispositivos suelen ser elevados y la compra y adquisición de los mismos es difícil, debiendo recurrir en ocasiones a terceros para poder adquirirlos, que elevan cuantiosamente el precio original del dispositivo.

Por otro lado, estos dispositivos no necesariamente poseen las especificaciones exactas que se requieren para una aplicación en particular o, por el contrario, aquellos que poseen

un valor más elevado poseen características que no son necesarias y no se ocuparán, haciendo uso de un dispositivo sobredimensionado para la aplicación en la que se lo emplea.

3 ESPECIFICACIONES

En las siguientes tablas se mencionan las especificaciones más importantes del dispositivo.

ESPECIFICACIONES GENERALES		
Frecuencia de muestreo	En conmutación de canales	Máximo 390 muestras/segundo
Resolución vertical	AVDD=5V 23 bits	$V_{RES} = \frac{5V}{2^{23}} = 0.596 \mu V$
Resolución libre de ruido	DATA RATE = 30000 SPS PGA = 1	NFR = 17.1 bits
	DATA RATE = 2.5 SPS PGA = 1	NFR = 23 bits
Cantidad de ruido	DATA RATE = 30000 SPS	100 < ppm < 1200
Linealidad	En el rango de 80 mV a 3 V. Factor de corrección de 1,0547	± 2,0514 %

ESPECIFICACIONES GENERALES (continuación)		
Cantidad de muestras pre-disparo	Cantidad de muestras post-disparo = 0	Máximo 39000 muestras
Cantidad de muestras post-disparo	Cantidad de muestras pre-disparo = 0	Máximo 39000 muestras

3.1 CARACTERÍSTICAS ELÉCTRICAS

Todas las especificaciones a -40°C a $+85^{\circ}\text{C}$, AVDD = + 5V, DVDD = + 3.3V, fCLKIN = 7.68MHz, PGA = 1

PARÁMETRO	CONDICIONES DE TESTEO	MÍNIMO	TÍPICO	MÁXIMO
Voltaje de entrada absoluto (AIN0-7, AINCOM a AGND)	Buffer off	AGND - 0.1 V	AVDD + 0.1 V	
	Buffer on	AGND	AVDD - 2.0 V	
Amplificador de ganancia programable		1	64	
Impedancia de entrada (AIN0-7, AINCOM a AGND)	Buffer off PGA = 1	$11 \text{ M}\Omega < Z_{in} < 81 \text{ M}\Omega$		
	Buffer on PGA = 1	$40 \text{ M}\Omega < Z_{in} < 62 \text{ M}\Omega$		
Data Rate	fCLKIN = 7.68MHz	2.5 SPS	30000 SPS	
Fuente de alimentación				
AVDD		4.75 V	5.25 V	
DVDD		1.8 V	3.6 V	
Corriente AVDD	Modo apagado	2 uA		
	Modo normal, PGA = 1, Buffer off	7 mA		
	Modo normal, PGA = 1, Buffer on	16 mA		
Corriente DVDD	Modo apagado	2 uA		
	Modo Standby, CLKOUT off, DVDD = 3.3V	95 uA		
	Modo Normal, CLKOUT off, DVDD = 3.3V	0.9 mA		
Disipación de potencia	Modo Normal, CLKOUT off, DVDD = 3.3V	38 mW		
	Modo Standby, DVDD = 3.3V	0.4 mW		

4 INGENIERÍA GENERAL

4.1 REQUISITOS MÍNIMOS

Req. #	Descripción del requisito
1	<p>El dispositivo debe tener una resolución temporal lo suficientemente elevada como para representar correctamente o de forma aceptable señales de frecuencias menores a los 15 Hz. Por lo tanto, tomando un mínimo de 10 muestras por ciclo, el dispositivo debe ser capaz de muestrear al menos cada 6 mS, es decir, a una frecuencia mínima de 166 muestras/segundo.</p>
2	<p>Debe poseer una resolución vertical, tomando como tensión de referencia 5 V, de al menos: $Res = \frac{5V}{2^{16}} = 76.294 \mu V$.</p>
3	<p>Los parámetros de: frecuencia de muestreo, nivel de disparo, canal de disparo, cantidad de muestras pre-disparo, cantidad de muestras post disparo y cantidad de archivos almacenables deben ser configurables por el usuario.</p>
4	<p>Las mediciones deben poseer consistencia temporal, ya sea a través de un ΔT constante entre cada muestra o registrando el tiempo en el que se toma cada muestra.</p>
5	<p>El código fuente generado debe ser liberado al cliente.</p>
6	<p>Cada archivo generado debe poseer un número, la fecha y la hora, las cuales deben mantenerse actualizadas, aunque el dispositivo no posea conexión a internet.</p>
7	<p>Debe poseer conexión remota, en el caso de que se encuentre conectado a internet. Se debe proveer una conexión en red local y no local y debe permitir la modificación de parámetros y la descarga de archivos.</p>
8	<p>El costo del dispositivo en su conjunto no debe superar los US\$ 150 al día de la fecha.</p>
9	<p>Los dispositivos que constituyen al conjunto deben ser de fácil accesibilidad en Argentina y tener la posibilidad de ser provistos a través de diversos proveedores.</p>
10	<p>Debe poseer mecanismos de protección, para evitar que el conversor A/D sufra daños ante niveles de la señal de entrada que sean excesivos o negativos.</p>

4.2 ALCANCES

En referencia a lo que el dispositivo final hará y lo que no, se deben aclarar los siguientes alcances y posibilidades brindadas por el mismo:

- El dispositivo tomará muestras a través de sensores especializados para la medición de vibraciones. Podrá hacerlo a través de hasta 8 canales de entrada analógicos de forma individual, es decir, un sensor por canal o 4 canales de forma diferencial usando una entrada como referencia y otra como medición.
- La frecuencia máxima de muestreo según las especificaciones es de 30000 muestras/segundo, pero es importante aclarar que esta frecuencia solo puede alcanzarse en el caso de trabajar con un solo canal. Esto es debido a que, si se deciden utilizar más canales, se deberán ir conmutando los sucesivos canales para medir de forma secuencial y no en forma paralela en el tiempo o instantánea. Por lo tanto, la frecuencia de muestreo final por cada canal si se decide utilizar más de un canal, será menor a la anteriormente especificada.
- Cuando las muestras tomadas superen un cierto valor umbral especificado por el usuario, se disparará el almacenamiento de las mediciones en un archivo de texto. Este archivo tendrá como nombre el número de archivo y la estampa de tiempo de cuando ocurrió el disparo. El formato será el siguiente:

“00XXXX-DD|MM|AAAAA-HH:MM:SS”

XXXX representa el número de archivo, cuya longitud variará a medida que se generen nuevos archivos.

- Se almacenarán en columnas las mediciones tomadas de los 8 canales y cada medición con su respectiva estampa de tiempo. La cantidad de muestras a almacenar por cada archivo será función de la cantidad “n” de muestras pre-disparo y las “m” muestras post disparo, parámetros seleccionables por el usuario. Por lo tanto, el dispositivo medirá constantemente y cuando alguna de las muestras supere el valor umbral especificado, se guardarán en el archivo las n muestras antes de dicho disparo y las m muestras posteriores al disparo.
- Las muestras se almacenarán en el formato combinacional arrojado de forma directa por el conversor A/D, y se incluirá en el encabezado del archivo la equivalencia para el pasaje a volts. También en dicho encabezado se incluirán los parámetros preseleccionados por el usuario.
- Los parámetros modificables por el usuario antes de realizar las mediciones se encontrarán en un archivo de configuración de texto. Estos parámetros son: frecuencia de muestreo, canal de disparo, nivel de disparo, cantidad de muestras pre-disparo, cantidad de muestras post disparo, cantidad de archivos almacenables. El canal de disparo hace referencia a aquel canal que, si alguna de las muestras

tomadas por él supera el valor umbral, disparará el almacenamiento en el archivo. Ningún otro canal no seleccionado provocará dicho disparo.

El nivel umbral de disparo se especifica en voltios. La cantidad de archivos almacenables indica la máxima cantidad de archivos que se almacenarán antes de empezar a sobrescribirse.

- Los archivos generados se almacenarán en la memoria del dispositivo hasta la cantidad especificada en cantidad de archivos almacenables. Si se supera dicha cantidad, los nuevos archivos irán sobrescribiendo a los archivos más antiguos.
- También se proveerá diversas formas de acceder al dispositivo de forma remota. Dentro de la red de área local, podrá accederse a la línea de comando de la Raspberry Pi a través del programa PuTTY, que funciona como un cliente SSH, al cual se le debe especificar la dirección IP y el puerto asignado a la placa. Si lo que se desea es acceder al escritorio de la Raspberry Pi, se utilizará el programa VNC Viewer, el cual nos permitirá acceder al dispositivo como si la conexión se realizara de forma directa al monitor y teclado de una computadora. Si se desea descargar los archivos y programas almacenados en el dispositivo se utilizará el programa WinSCP, que es un software específico de transferencia de archivos.
- Por último, si se desea acceder al dispositivo de forma remota desde cualquier lugar, sin importar si nos encontramos o no en la misma red a la que se encuentra conectada la placa Raspberry Pi, lo haremos a través del servicio de mensajería de Telegram. A través del mismo, se le presentará al usuario un menú de opciones por el cual podrá seleccionar la acción que deseé realizar. Las mismas incluyen la descarga de archivos, mostrar el listado de archivos, ver los parámetros actuales, modificar dichos parámetros y testear la temperatura de la placa. De esta forma podrá manejar el dispositivo desde su celular de forma muy cómoda e intuitiva.

4.3 FUNCIONAMIENTO GENERAL

El funcionamiento general del dispositivo es el siguiente:

- Conexionado

La placa conversora AD/DA se encuentra montada sobre la placa Raspberry Pi, por lo tanto, las conexiones necesarias entre ambas placas ya se encuentran realizadas. A la placa conversora se conecta el circuito de protección. Este circuito es el encargado de limitar los niveles de las señales de entrada para evitar que superen el rango máximo admisible por las entradas analógicas de la placa conversora. Evita tanto que los valores de entrada superen el nivel máximo permitido, el cual depende de la tensión de alimentación externa, como también evita que la señal de entrada supere un cierto nivel de tensión negativa, los cuales podrían dañar o incluso causar la destrucción del conversor A/D.

Por lo anteriormente mencionado, el usuario sólo debe conectar las salidas de los sensores de vibración a las entradas correspondientes del circuito de protección de entrada. Las entradas están configuradas de forma tal que cada entrada analógica sea un canal de medición y la entrada AINCOM sea el terminal de referencia. Este es el modo llamado “entrada o terminal único”, el cual permite aprovechar los 8 canales de medición del conversor. Desde el programa puede elegirse el “modo diferencial”, que permite tomar una entrada como canal de medición y otra entrada como referencia, reduciéndose la cantidad total de canales de medición a 4.

La placa Raspberry Pi 3 B+, toma su alimentación a través de un adaptador 220VAC/5VDC conectado a la red eléctrica.

Si se desea tener conexión a internet, esta puede obtenerse a través de un cable ethernet conectado de forma directa a la Raspberry Pi o a través de conexión Wi-Fi una vez que el sistema operativo haya iniciado.

- Medición

Una vez iniciada la Raspberry Pi, automáticamente se ejecutan los dos programas desarrollados. Uno es el encargado de realizar las mediciones y el otro es el encargado de comunicar la placa con el usuario de forma remota a través de Telegram.

El programa de medición realiza dos acciones principales. La primera es medir de forma continua los 8 canales, comutando cada uno de ellos desde el canal 0 al 7. Toma la muestra de un canal, la almacena en una posición de la primera columna de un arreglo bidimensional y comuta al canal siguiente para tomar la muestra correspondiente. Luego obtiene el tiempo en el que tomó dichas muestras y lo almacena en la última columna del arreglo. De esta forma se obtiene un arreglo que posee 8 mediciones, donde cada columna representa la muestra obtenida de cada canal, y en la novena columna se tiene la estampa de tiempo correspondiente. Luego se sigue con la siguiente medición almacenando las nuevas muestras en filas sucesivas.

Esta medición continua de canales la realiza en forma ininterrumpida mientras no se supere un determinado nivel umbral asignado por el usuario. Si alguna de las muestras obtenidas por el canal seleccionado supera este nivel umbral, el cual el usuario puede modificar a gusto, así como también el canal cuya muestra debe evaluarse, se dispara el guardado automático de las muestras obtenidas dentro de un archivo de texto. El archivo generado posee en su nombre un determinado número y la fecha, hora, minuto y segundo exacto en la que se produjo el disparo. Dentro de este archivo se especifican los parámetros seleccionados por el usuario, el factor por el cual deben multiplicarse las muestras para obtener su valor en volts, y las muestras y estampas de tiempo propiamente dichas.

Además de la muestra que desencadena el guardado de las muestras, se almacenan “n” muestras anteriores al disparo y “m” muestras posteriores a él, siendo n y m parámetros también seleccionables por el usuario.

Una vez que finaliza el guardado de las muestras en el archivo, se retoma nuevamente la tarea de medir constantemente los canales de entrada.

- Manejo de archivos

Debido a que la memoria de la Raspberry Pi es limitada, la generación de archivos debe estar controlada para evitar que se alcance el límite máximo de memoria. Para lograr este control se utiliza un sexto parámetro modificable por el usuario llamado “cantidad de archivos almacenables”. Este parámetro es ingresado por el usuario e indica la cantidad máxima de archivos que se generarán antes de comenzar a sobre escribirse. Por lo tanto, una vez alcanzado dicho número de archivo, los nuevos archivos generados sobrescribirán a los más antiguos. De esta forma limitamos la ocupación de memoria.

- Parámetros modificables por el usuario

Se han mencionado ciertos parámetros que le permiten al usuario controlar ciertos aspectos y límites a la hora de realizar las mediciones y generar los archivos. Estos parámetros son los siguientes:

- 1) Canal de disparo:

Con este parámetro el usuario puede seleccionar cuál de los 8 canales de medición provocará el guardado de las muestras tomadas en un nuevo archivo, si se supera el valor umbral especificado en nivel de disparo. Los canales se enumeran del 0 al 7.

- 2) Frecuencia de muestreo:

Al modificar la frecuencia de muestreo se modifica la velocidad con la que el dispositivo toma las sucesivas muestras. A mayor frecuencia de muestreo, mayor velocidad en la toma de mediciones.

Existe un amplio abanico de posibles frecuencias de muestreo seleccionables desde un mínimo de 2.5 SPS (Samples Per Second), hasta un máximo de 30000 SPS. Es importante aclarar que, aunque estas sean las frecuencias de muestreo especificadas en la hoja de datos del conversor ADS1256, la frecuencia real de muestreo es mucho menor (Véase apartado 1), debido a la conmutación de canales, los tiempos de espera necesarios para que el conversor pueda tomar muestras de forma correcta, etc.

3) *Nivel de disparo:*

Este es el valor umbral que es comparado constantemente con la muestra del canal de disparo. En caso de que alguna muestra supere este valor umbral, se detiene la medición de los canales y se guardan las muestras en el archivo.

Se ingresa en el siguiente formato:

“U.dc” en volts.

Por ejemplo: 1.38 V

4) *Cantidad de muestras post-disparo:*

Una vez superado el nivel de disparo, el dispositivo seguirá tomando una determinada cantidad de muestras. Esta cantidad es la especificada en este parámetro. De esta forma si el usuario desea 100 muestras luego del disparo, el dispositivo utiliza este parámetro para tomar 100 muestras más de todos los canales antes de realizar el guardado en el archivo correspondiente.

5) *Cantidad de muestras pre-disparo:*

De la misma forma en que pueden almacenarse muestras posteriores al disparo, pueden almacenarse muestras anteriores a este. Esto puede hacerse debido a que el dispositivo mide constantemente y almacena las muestras en un buffer que es el arreglo bidimensional ya mencionado. De esta manera, si el usuario desea 100 muestras anteriores al disparo, ya se encuentran almacenadas. En caso de que el disparo se produzca al poco tiempo de haber iniciado el dispositivo y si no se poseen muestras anteriores suficientes para llegar a la cantidad especificada, las muestras faltantes toman el valor 0.

6) *Cantidad de archivos almacenables:*

Este parámetro ya fue explicado anteriormente, pero se utiliza como la cantidad máxima de archivos que se generarán antes de comenzar a sobrescribir a los archivos más nuevos. De esta forma, si el usuario desea que se almacenen solo 100 archivos, una vez llegado al archivo número 102 (debido a que el primer archivo es el número 0), este nuevo archivo tomará el valor ‘000’ y sobrescribirá al archivo más antiguo, reiniciándose el conteo de estos.

Estos parámetros están contenidos en un archivo de configuración de texto. El usuario puede abrir este archivo y modificar el parámetro que desee, o acceder a dichos parámetros de forma remota. Los mismos poseen el siguiente formato:

1-Canal-4

2-Frecuencia-30000

3-Nivel-1.57

4-Muestras posttrigger-12000

5-Muestras pretrigger-673
6-Cantidad de archivos-1000

#Consideraciones#

*La suma de la cantidad de muestras pre y post disparo no debe superar las 39000 muestras.

*La frecuencia de muestreo real solo se aproxima a la seleccionada en el caso de leer un solo canal. Para 8 canales la frecuencia máxima es de aproximadamente 390 muestras/segundo.

*El nivel máximo admisible en las entradas analógicas no debe superar los 5 V con el buffer de entrada deshabilitado, por lo que el nivel de disparo debe ser menor.

*Los canales de lectura van del número 0 al número 7.

El programa lee y obtiene estos parámetros al inicio de este, pero los mismos pueden ser aplicados nuevamente de forma remota a través de una de las opciones ofrecidas por la interfaz de Telegram.

Es importante aclarar que la elección de contener los parámetros de configuración en un archivo implica la gran ventaja de que, en caso de que el dispositivo se quede sin alimentación, al reiniciar no requiere que exista un usuario presente en el lugar que deba ingresar los parámetros en forma manual, directamente los vuelve a obtener de este archivo.

- Acceso remoto:

Los archivos y los parámetros de configuración pueden accederse tanto de forma directa a la placa como de forma remota a ella. Para el acceso remoto se han planteado diversas alternativas.

- Acceso remoto desde dentro de la misma red:

Si se accede al dispositivo desde una computadora cuya dirección IP se encuentre dentro de la misma red que la dirección IP del dispositivo, puede realizarse desde 3 programas de software distintos. Si solo se desea acceder a la línea de comando de la Raspberry Pi, se utilizará el software PuTTY. Si lo que se desea es la descarga de los archivos emplearemos el software WinSCP. Por último, si desea acceder al escritorio de la placa, en forma tal de usar el monitor, mouse y teclado de la computadora como si estos estuvieran conectados de forma directa a la Raspberry Pi, se usará el programa VNC Viewer.

- Acceso remoto desde fuera de la red:

Si desea acceder al dispositivo desde cualquier computadora o dispositivo, independientemente de si esta se encuentra conectada a la misma red que el dispositivo, puede hacerse desde el servicio de mensajería de Telegram. A través de este servicio, se le presenta una serie de opciones y menús al usuario que le permitirá de forma muy cómoda e intuitiva modificar los parámetros de configuración, listar y descargar los archivos, entre otras opciones (véase apartado 2).

4.4 DIAGRAMA EN BLOQUES

A continuación, se muestra un diagrama en bloques generalizado del dispositivo y sus componentes principales:

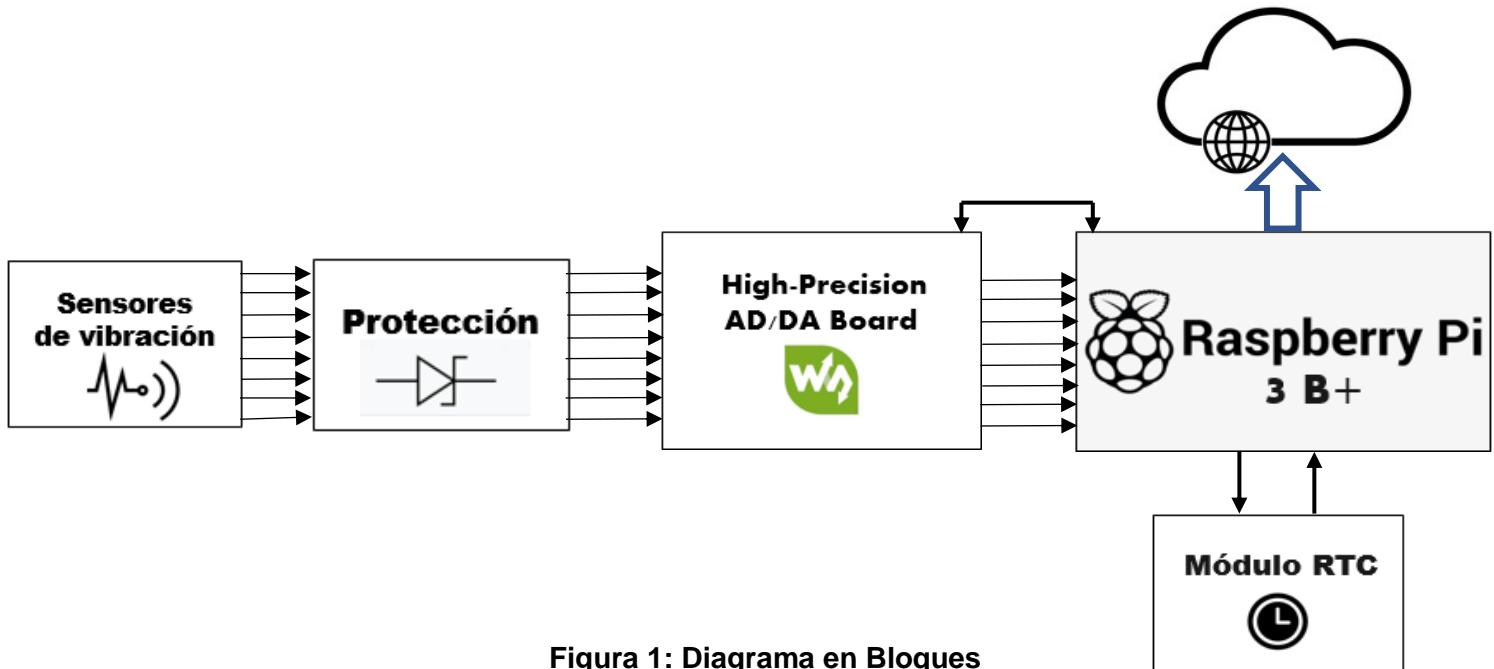


Figura 1: Diagrama en Bloques

- Sensores de vibración: este bloque representa a los 8 transductores que se conectarán a las entradas del bloque conversor. Estos sensores de vibración constan de acelerómetros de altísima precisión, aunque perfectamente pueden conectarse cualquier tipo de sensor, y serán los encargados de captar las vibraciones de las estructuras analizadas y entregar una señal eléctrica acorde a la magnitud y frecuencia de las vibraciones producidas.
- Protección: este segundo bloque representa al circuito encargado de proteger a la placa conversora. El mismo está compuesto de un arreglo de diodos que limitan y mantienen las señales entregadas por los sensores dentro de los rangos admisibles para las entradas analógicas del conversor A/D.

- Placa conversora analógica/digital-digital/analógica de alta precisión: la placa conversora es la encargada de recibir en sus entradas analógicas las señales eléctricas provenientes de los transductores de vibración y convertir dicha señal analógica en una señal digital correspondiente.

Esta placa conversora posee dos circuitos integrados principales. El primero es el ADS1256, integrado que realiza la conversión de señales analógicas en señales digitales. Posee una resolución de 24 bits, lo que lo convierte en un conversor de gran precisión, y una frecuencia de muestreo máxima de 30.000 muestras/segundo. El otro circuito integrado de interés es el DAC8532, que permite a partir de una señal digital obtener una señal analógica de salida. Posee 16 bits de resolución. De los dos, se hará uso en su gran mayoría del primero, ya que el fin del dispositivo en su conjunto es la medición de vibraciones. Sin embargo, es importante conocer las posibilidades que nos brinda esta placa para futuras aplicaciones.

- Raspberry Pi modelo 3 B+: este bloque representa al cerebro del dispositivo. La placa Raspberry Pi es un ordenador de placa reducida de bajo costo. La misma cuenta con el sistema operativo Raspbian e implementa los programas principales y secundarios necesarios para el funcionamiento de todo el dispositivo.

El programa de medición recibe las señales digitales del conversor A/D y las almacena de la forma explicada anteriormente. De la misma forma, es la placa Raspberry Pi la que, mediante el programa y librerías dedicadas, le ordena al conversor que tome muestras y las envíe. De la misma forma, el conversor le indica cuando se encuentra listo para tomar la siguiente muestra. El programa de acceso remoto permite la comunicación, el paso de directivas y la descarga de los archivos a través del servicio de mensajería de Telegram.

También es en la memoria de este dispositivo donde se almacenan los archivos generados con las muestras tomadas.

- Acceso a internet: la placa Raspberry Pi tiene la posibilidad de acceder a internet, ya sea a través de un cable Ethernet, como también a través de Wi-Fi. Cuando posee acceso, la fecha y la hora del dispositivo se actualiza de forma automática a través de internet.
- Módulo RTC: una de las limitaciones que posee Raspberry Pi es la carencia de un reloj interno de tiempo real. Esto implica que, si la placa no posee acceso a internet, al iniciar la misma, la fecha y la hora no son actualizadas de forma automática, sino que poseen la fecha y hora de la última vez que fue iniciada. Debido a que tanto las muestras, como el nombre del archivo deben poseer la hora exacta en la que se produjeron las vibraciones, se hizo necesario el uso de un dispositivo que, sin importar si la Raspberry Pi poseía conexión a internet o si se encontraba sin alimentación, mantuviera de forma ininterrumpida la hora actualizada. Este es el propósito de este módulo, por el cual la Raspberry Pi obtiene la fecha y la hora siempre actuales.

5 INGENIERÍA DE DETALLE

5.1 HARDWARE

Comenzaremos mostrando en detalle cada uno de los dispositivos utilizados, su configuración de pines, especificaciones, etc.

5.1.1 Raspberry Pi 3 B+:

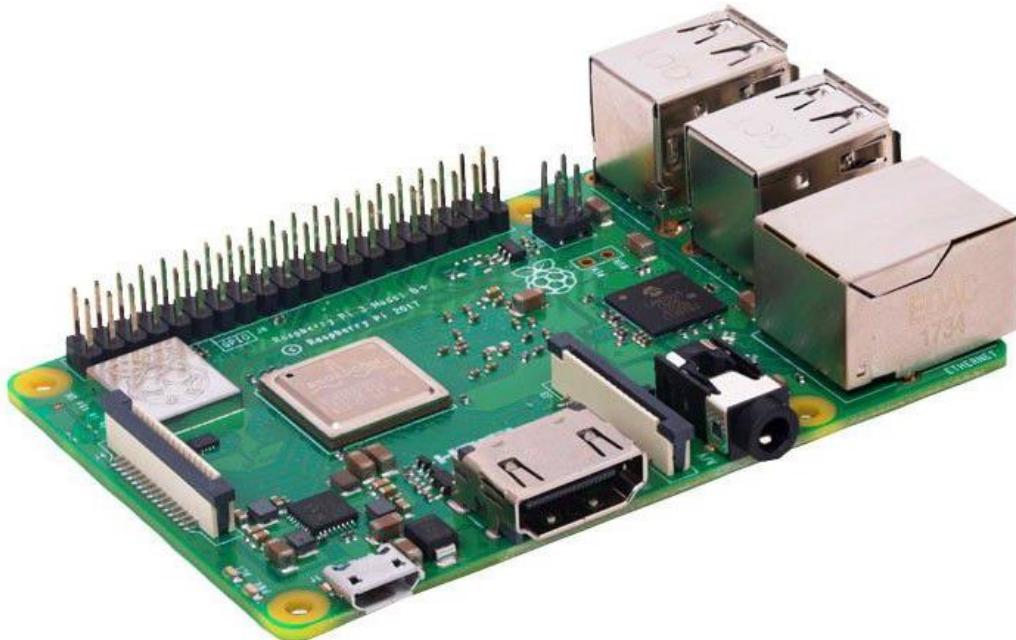


Figura 2: Vista real de la placa Raspberry Pi 3 B+.

Especificaciones técnicas:

- CPU + GPU: **Broadcom BCM2837B0**, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: **1GB LPDDR2 SDRAM**
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- GPIO de 40 pines
- HDMI
- 4 puertos USB 2.0
- Puerto CSI para conectar una cámara.
- Puerto DSI para conectar una pantalla táctil
- Salida de audio estéreo y vídeo compuesto
- Micro-SD
- Power-over-Ethernet (PoE)

Configuración de pines:

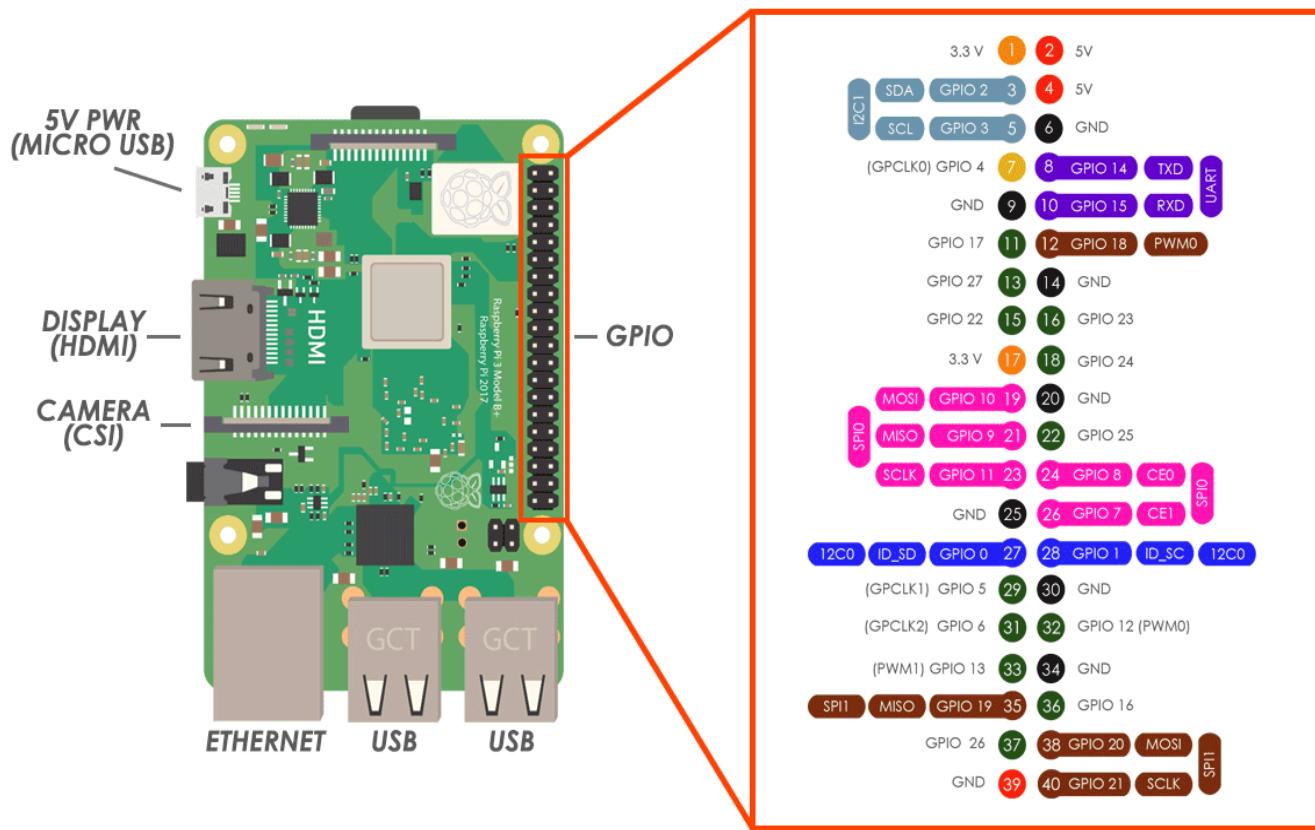


Figura 3: Configuración de pines de la Raspberry Pi.

La alimentación es suministrada con un cargador de 5 V y 5.1 A, a través del conector micro USB.

Para el manejo de esta placa puede conectarse un teclado, mouse y monitor, gracias a sus 4 puertos USB y su conector HDMI.

También puede accederse a internet a través de Ethernet, ya que posee un puerto RJ45.

5.1.2 Placa conversora AD/DA de alta precisión:

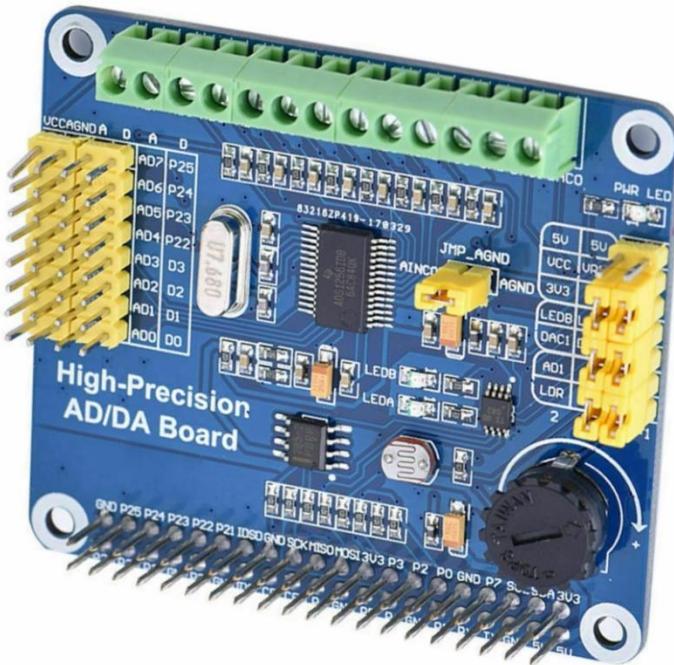


Figura 4: Vista real de la placa conversora AD/DA.

Características técnicas:

- ADS1256 integrado, ADC de alta precisión de 8 canales y 24 bits (entrada diferencial de 4 canales), frecuencia de muestreo de 30 ksps.
- DAC8532 integrado, DAC de alta precisión de 2 canales y 16 bits.
- Interfaz de entrada integrada mediante cabezales de pines, para conectar señales analógicas:
- El pinout es compatible con el estándar de interfaz del sensor Waveshare, fácil de conectar varios módulos de sensores analógicos.
- Interfaz de entrada/salida incorporada a través de terminales de tornillo, para conectar señales analógicas / digitales.
- Cuenta con circuito de detección AD/DA.

Configuración de pines:

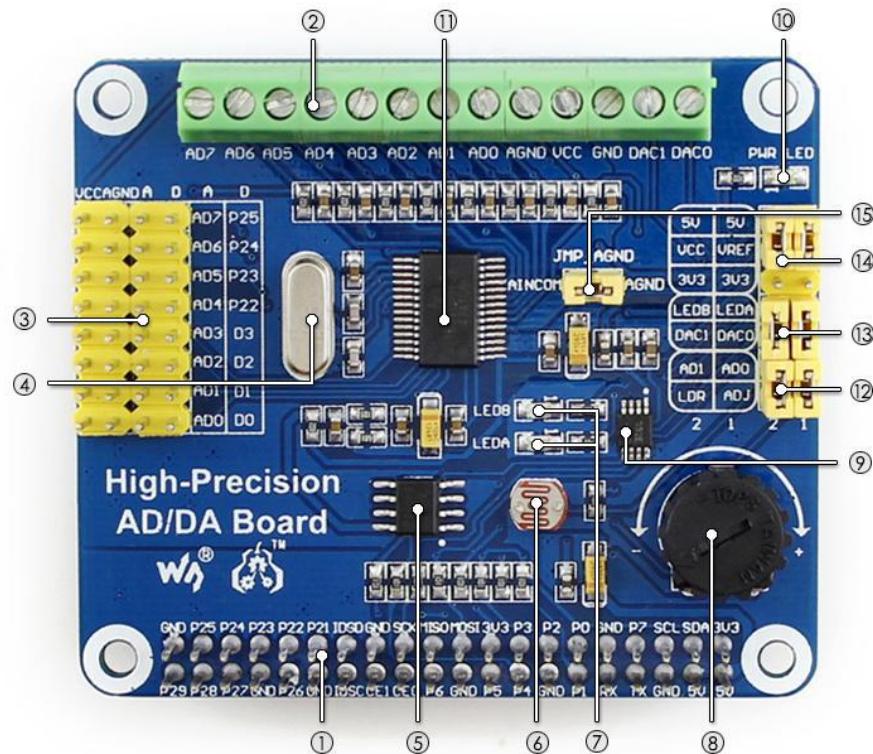


Figura 5: Configuración de pines de la placa conversora.

1. **Interfaz GPIO de Raspberry Pi:** para conexión con la placa Raspberry Pi.
2. **Entrada / salida AD / DA:** terminales de tornillo.
3. **Entrada AD:** cabezales de pines; los pines de salida son compatibles con el estándar de interfaz del sensor Waveshare, fácil para conectar varios módulos de sensores analógicos.
4. **Cristal de 7.68 MHz.**
5. **LM285-2.5:** proporciona voltaje de referencia para el chip ADC.
6. **Foto resistor.**
7. **indicador de salida LED.**
8. **Potenciómetro de 10 KΩ.**
9. **DAC8532:** DAC de alta precisión de 16 bits, 2 canales.
10. **Indicador de alimentación.**
11. **ADS1256:** ADC de alta precisión de 24 bits, 8 canales (entrada diferencial de 4 canales)
12. **Puente de prueba ADC.**
13. **Puente de prueba DAC.**

14. Puente de selección de potencia.

15. Configuración de tierra de referencia de ADC: cuando se ingresa una sola entrada analógica, AINCOM es el terminal de referencia, el cual se puede conectar a GND o a un voltaje de referencia externo.

1) Entrada / salida AD / DA

AD0-AD7: entradas analógicas

AGND: tierra analógica

GND: tierra digital

VCC: fuente de alimentación (3.3V y 5V opcional, se puede cambiar configurando el puente de selección de energía)

DA0-DA1: Salida analógica

2) Entrada analógica

AD0-AD7: entrada analógica ADS1256

D0-D3: GPIO de ADS1256

P22-P25: GPIO de Raspberry Pi

AGND: tierra analógica

3) LDR: resistencia dependiente de la luz incidente, es decir, una fotorresistencia.

Al conectar el puente entre AD1 y LDR, la MCU puede leer la salida de voltaje de LDR desde el pin AD1.

4) LEDA / LEDB: indicador de salida LED.

Al conectar el puente entre LEDA / LEDB y DAC0 / DAC1, la salida de voltaje de DAC0 / DAC1 se puede estimar aproximadamente por el brillo de LEDA / LEDB.

5) ADJ: La resistencia ajustable del potenciómetro.

Al conectar el puente entre AD0 y ADJ, la MCU puede leer la salida de voltaje del potenciómetro del pin AD0.

6) LED PWR: indicador de encendido.

7) Puente de selección de potencia.

VCC: selección de fuente de alimentación

VREF: voltaje de entrada de referencia

3V3: salida de 3.3V

5V: salida de 5V

8) JMP_AGND: Puente de tierra analógico.

Para mediciones de un solo extremo, utilizar AINCOM (común de entrada analógica) como entrada común, que se puede conectar a AGND o a un voltaje de referencia externo. Para medidas diferenciales, no utilizar AINCOM.

- **ADS1256**

Dentro de la placa conversora, debemos prestar especial atención al circuito integrado principal encargado de realizar la conversión analógica a digital.

Diagrama en bloques:

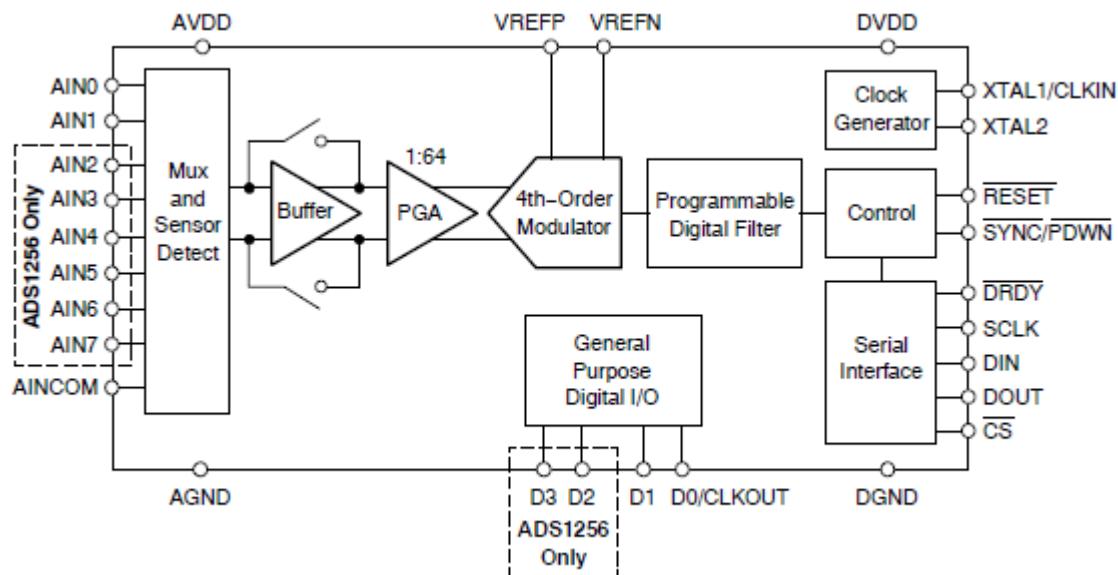


Figura 6: Diagrama en bloques del conversor ADS1256.

Especificaciones técnicas:

- 24 bits.
- Resolución sin ruido de hasta 23 bits.
- $\pm 0.0010\%$ de no linealidad (máx.).
- Tasas de salida de datos a 30 kSPS
- Ciclo rápido de canal:
-18.6 bits sin ruido (21.3 bits efectivos) a 1.45 kHz.
- Conversiones de una sola vez en configuración de ciclo único
- Multiplexor de entrada flexible con detección de sensor:
 - Cuatro entradas diferenciales (solo ADS1256).
 - Ocho entradas de un solo extremo (solo ADS1256).
- Búfer de entrada estabilizado por chopper.
- PGA de bajo ruido: ruido de entrada de 27 nV.
- Auto calibración y calibración del sistema para todos los ajustes de PGA.

- Interfaz serial compatible con SPI™ tolerante con 5 V.
- Suministro analógico: 5 V
- Suministro digital: 1.8V a 3.6V
- Disipación de energía:
 - Tan bajo como 38 mW en modo normal.
 - 0.4 mW en modo de espera (standby).

ABSOLUTE MAXIMUM RATINGS

over operating free-air temperature range unless otherwise noted⁽¹⁾

		ADS1255, ADS1256	UNIT
AVDD to AGND		-0.3 to +6	V
DVDD to DGND		-0.3 to +3.6	V
AGND to DGND		-0.3 to +0.3	V
Input Current		100, Momentary	mA
		10, Continuous	mA
Analog inputs to AGND		-0.3 to AVDD + 0.3	V
Digital inputs	DIN, SCLK, CS, RESET, SYNC/PDWN, XTAL1/CLKIN to DGND	-0.3 to +6	V
	D0/CLKOUT, D1, D2, D3 to DGND	-0.3 to DVDD + 0.3	V
Maximum Junction Temperature		+150	°C
Operating Temperature Range		-40 to +105	°C
Storage Temperature Range		-60 to +150	°C
Lead Temperature (soldering, 10s)		+300	°C
...			

Estos rangos máximos ya son tenidos en cuenta dentro de la placa conversora, por lo que solo debemos tener en cuenta los rangos máximos de las entradas analógicas.

Consideración especial:

La máxima frecuencia de muestreo se obtiene al colocar un cristal externo al circuito integrado de 7.68 MHz, que ya viene incluido en la placa conversora, por lo tanto, la máxima frecuencia de muestreo a la salida será de 30000 muestras/segundo. Esta velocidad se consigue solo si se lee un solo canal. Si se desean leer los 8 canales, la commutación del multiplexor a la entrada del ADC provoca retrasos que producen un decaimiento de esta frecuencia máxima a unas 4374 muestras por segundo. Sin embargo, entre medición y medición el conversor requiere un cierto tiempo para estabilizarse antes de realizar un nuevo muestreo. Este tiempo adicional provoca que la frecuencia máxima de muestreo final sea de aproximadamente 390 muestras/segundo. Obviar este tiempo implicaría obtener mediciones erróneas. Además, la frecuencia de salida es función de la ganancia asignada al conversor, pues para cada configuración del PGA (amplificador de ganancia controlada) se intercalan en la entrada distintas capacidades que influyen en la

frecuencia final de muestreo. También depende de la cantidad de promediaciones que se realicen, las cuales son configurables.

5.1.3 Circuito de protección:

Puede observarse en el diagrama de bloques del ADS1256 que luego del multiplexor de entrada existe un buffer de tensión. El mismo puede habilitarse o deshabilitarse a través de instrucciones en el programa de medición. La habilitación de este buffer incrementa significativamente las impedancias de entrada de los canales analógicos, permitiendo acercarse a la situación ideal de impedancia de entrada infinita comparada a la impedancia de salida de los sensores. Esto trae la ventaja de obtener tensiones de entrada más cercanas a las deseadas.

Sin embargo, como contrapartida se reduce el rango efectivo de los niveles de señal en las entradas analógicas:

Voltaje de entrada absoluto (AIN0-7, AINCOM a AGND)	Buffer off	AGND – 0.1 V	AVDD + 0.1 V
	Buffer on	AGND	AVDD – 2.0 V

Para proteger las entradas analógicas en ambas situaciones, se diseñó un circuito muy simple que limita a los valores máximos de entrada las señales provenientes de los transductores.

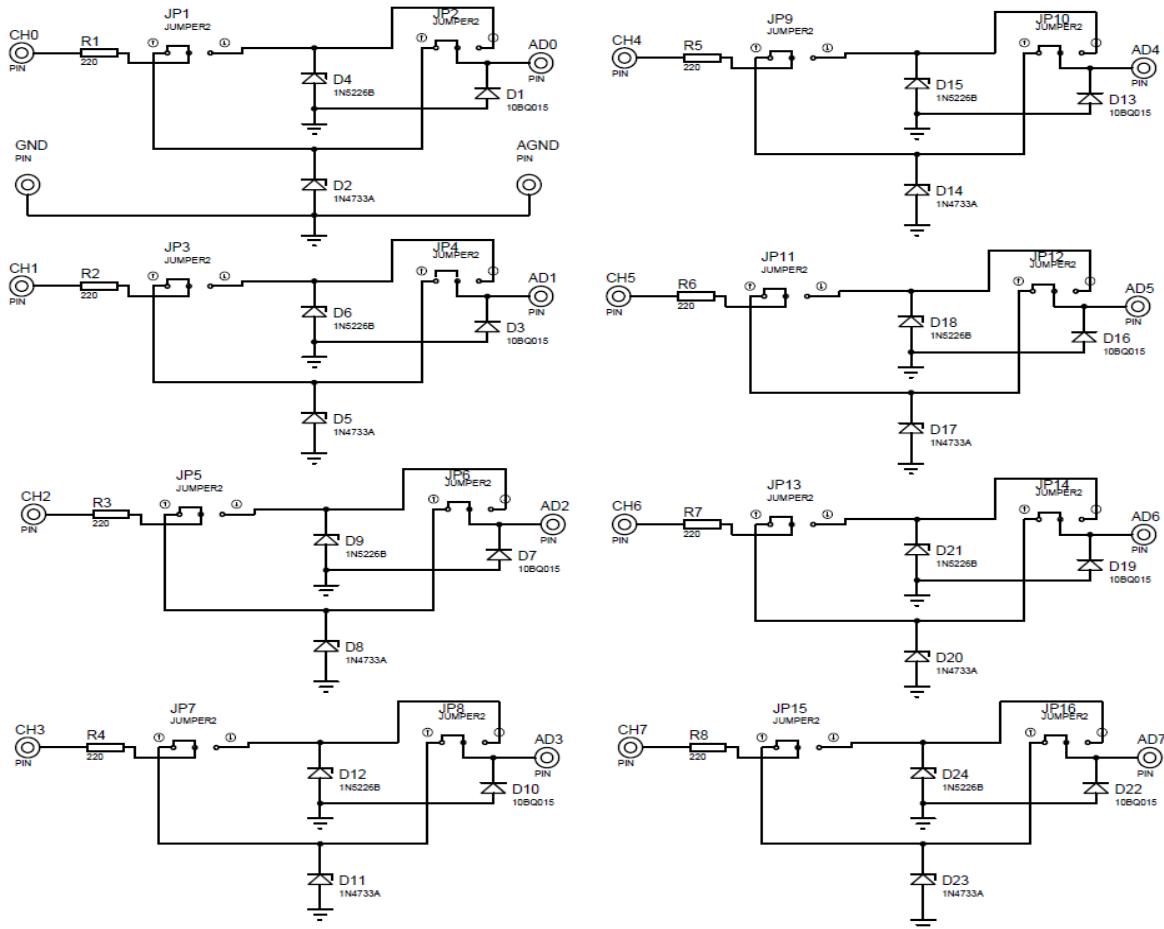


Figura 7: Circuito esquemático de la placa de protección.

Puede observarse que, a través de dos jumpers, uno a la entrada y uno a la salida, puede elegirse el rango de entrada para cada canal de entrada, para el caso en que el buffer se encuentre habilitado o que se encuentre deshabilitado.

Una tabla de verdad con las 4 posiciones posibles de los jumpers se muestra a continuación:

Posición	Jumper 1	Jumper 2	Límite superior	Límite inferior
Posición 1	SI	SI	5.1 V	-0.19 V
Posición 2	NO	NO		
Posición 1	SI	NO	—	—
Posición 2	NO	SI		
Posición 1	NO	SI	—	—
Posición 2	SI	NO		

Posición 1	NO	NO	3.3 V	-0.19 V
Posición 2	SI	SI		

Donde:

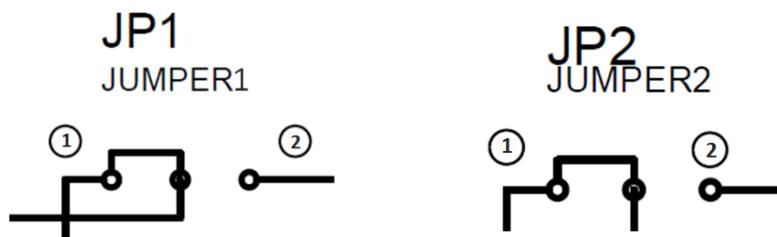


Figura 8: Posiciones de los jumpers del circuito de protección.

Cuando los jumpers se encuentran en posiciones distintas entre sí (representado con ‘—’ en el cuadro), la señal aplicada a la entrada desde los sensores queda desconectada de las entradas del conversor, ya que se conecta la entrada de uno de los diodos de protección con la salida del otro diodo. Sólo cuando ambos jumpers se encuentren en la misma posición, es decir, ambos en la posición 1 o ambos en la posición 2, las señales de los sensores se conectarán a las entradas del conversor, previo a pasar por el limitador de 5.1 V o el de 3.3 V respectivamente.

El mismo fue simulado en el software Multisim, obteniendo los siguientes resultados:

- Sin buffer habilitado:

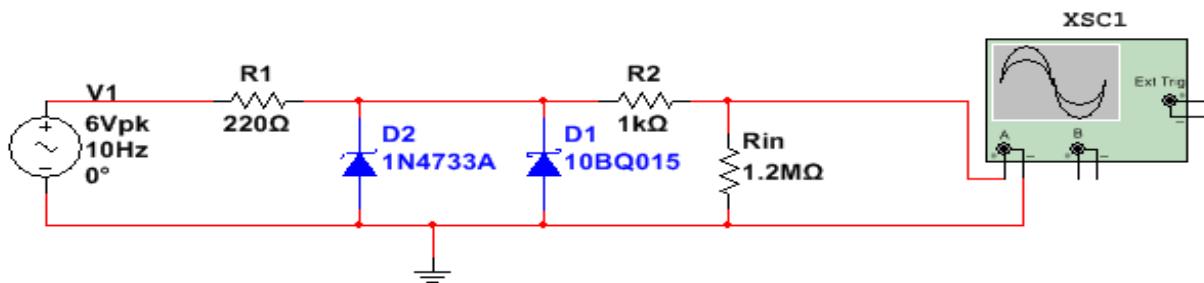


Figura 9: Simulación del circuito de protección con buffer de entrada deshabilitado.

Se ha colocado una impedancia de entrada de $1.2\text{ M}\Omega$, impedancia menor a cualquiera de las impedancias de entrada medidas, pero nos permite considerar el peor caso. También se ha tenido en cuenta la resistencia de $1\text{ k}\Omega$ entre el terminal de entrada y el ADS1256 colocado en la placa conversora.

Cuando el buffer de tensión de entrada se encuentra deshabilitado, el rango de entrada se encuentra entre -0.1 V y 5.1 V. Por esto, se decidió colocar el diodo Zener 1N4733A que posee las siguientes características:

Type	Zener Voltage Range ^{3), 5)}			Maximum Zener Impedance ¹⁾ at I _{ZK}			Maximum Reverse Leakage Current		Maximum Surge Current ⁴⁾ at T _a = 25 °C	Maximum Regulator Current ²⁾
	V _{Znom}	V _Z	I _{ZT}	r _{ZJT}	r _{ZJK}	mA	I _R	at V _R	I _{ZSM} (mA)	I _{ZM} (mA)
	V	V	mA	Ω	Ω	mA	μA	V		
1N4727A	3	2.85...3.15	83	10	400	1	150	1	1375	275
1N4728A	3.3	3.13...3.47	76	10	400	1	150	1	1375	275
1N4729A	3.6	3.42...3.78	69	10	400	1	100	1	1260	252
1N4730A	3.9	3.7...4.1	64	9	400	1	100	1	1190	234
1N4731A	4.3	4.08...4.52	58	9	400	1	50	1	1070	217
1N4732A	4.7	4.46...4.94	53	8	500	1	10	1	970	193
1N4733A	5.1	4.84...5.36	49	7	550	1	10	1	890	178
1N4734A	5.6	5.32...5.88	45	5	600	1	10	2	810	162
1N4735A	6.2	5.89...6.51	41	2	700	1	10	3	730	146

Este diodo Zener limitará la tensión máxima positiva a 5.1 V.

Para la limitación de la tensión negativa se decidió colocar el diodo Schottky 10BQ015, el cual posee las siguientes características:

Characteristics	10BQ015	Units
I _{F(AV)} Rectangular waveform	1.0	A
V _{RRM}	15	V
I _{FSM} @ t _p = 5μs sine	140	A
V _F @ 1.0Apk, T _J = 75°C	0.30	V
T _J	-55 to 100	°C

Las simulaciones realizadas arrojaron lo siguiente:

- Sin superar los 5.1 V:

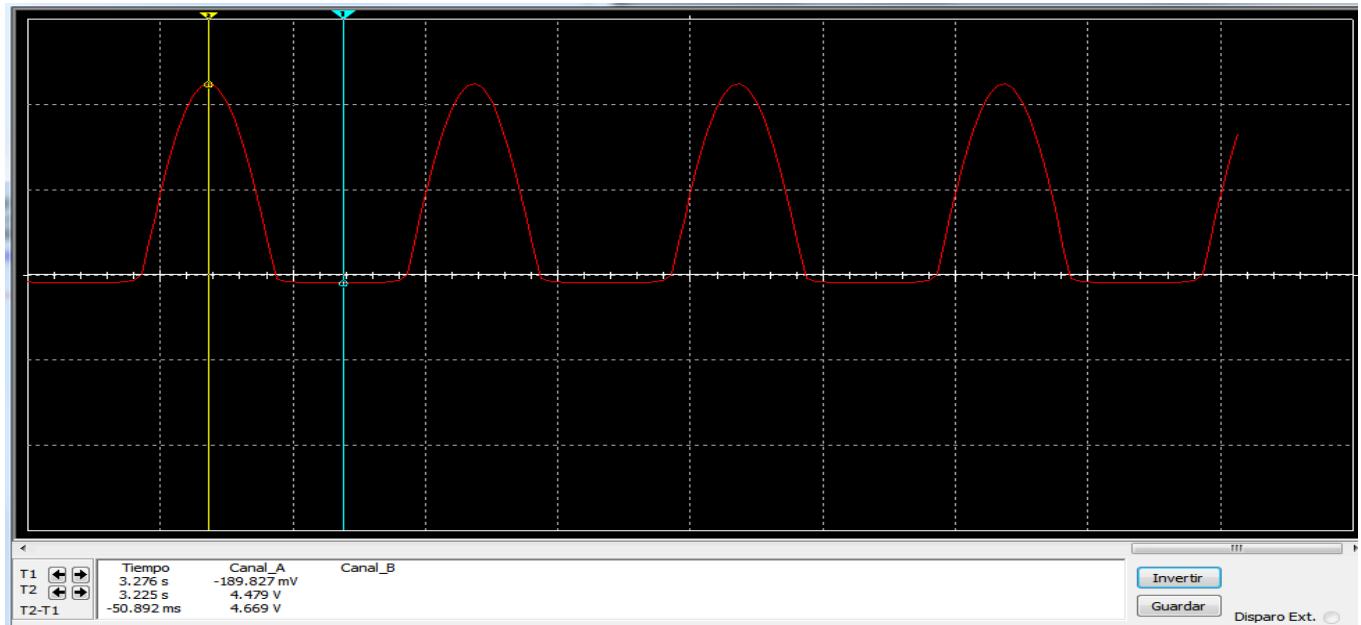


Figura 10: Simulación de la señal obtenida sin superación del rango máximo con buffer deshabilitado.

Puede observarse que el pico de tensión máximo alcanza los 4.48 V sin distorsión (se colocó una señal senoidal de 4.5 V pico y 10 Hz). Por el contrario, el semiciclo negativo se ve recortado a -0.189 V.

- Superando los 5.1 V:

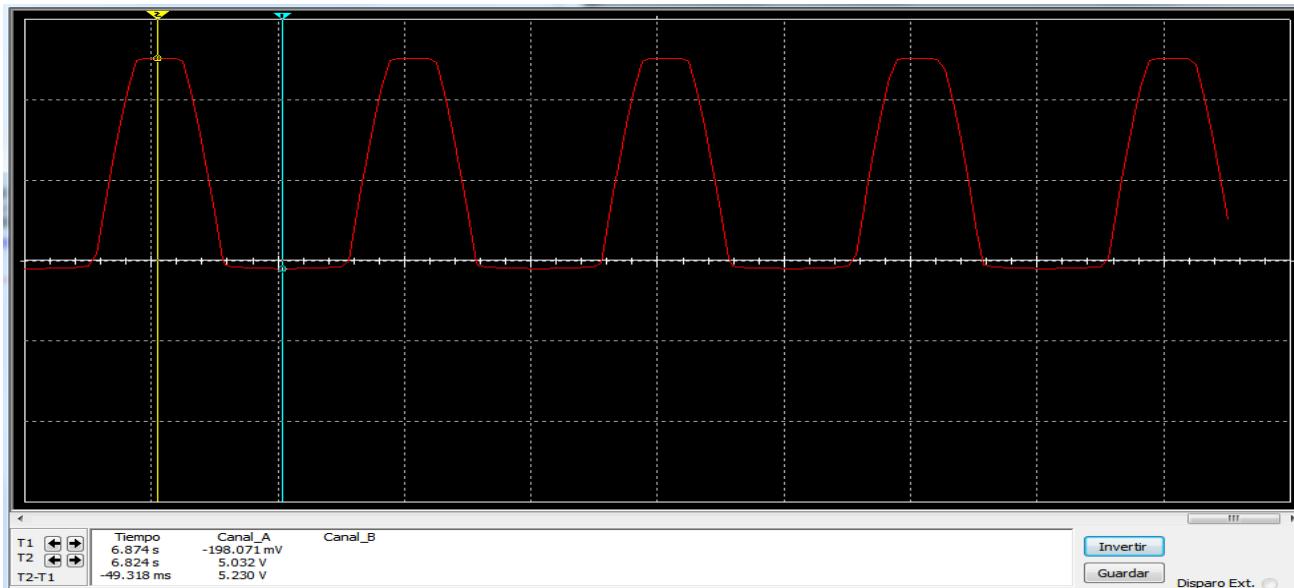


Figura 11: Simulación de la señal obtenida con superación del rango máximo con buffer deshabilitado.

En cambio, cuando la señal de entrada supera los 5 V, el pico positivo se ve recortado a los 5.03 V. La distorsión de la señal comienza aproximadamente en los 4.9 V.

- Con el buffer habilitado:

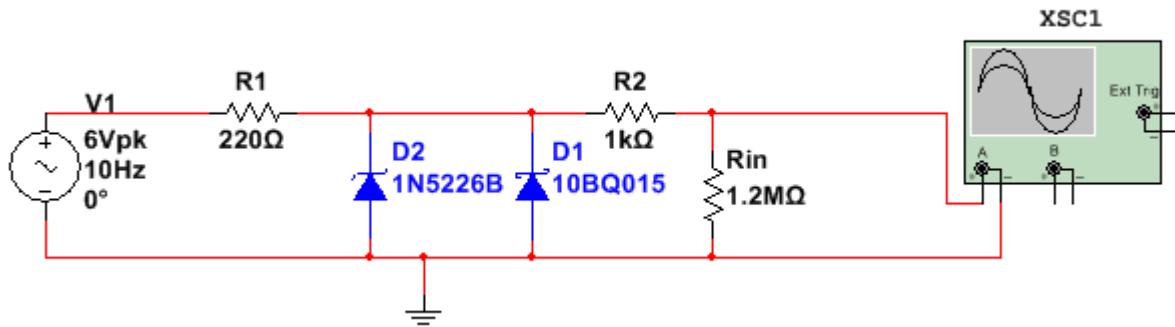


Figura 12: Simulación del circuito de protección con buffer de entrada habilitado.

Cuando habilitamos el buffer de tensión de la entrada, el rango de entrada se encuentra entre 0 V y 3 V. Por esto, se decidió colocar el diodo Zener 1N5226B que limita la tensión máxima a 3.3 V.

ELECTRICAL CHARACTERISTICS ($T_{amb} = 25^{\circ}\text{C}$, unless otherwise specified)							
PART NUMBER	ZENER VOLTAGE RANGE (I ^t)	TEST CURRENT		REVERSE LEAKAGE CURRENT		DYNAMIC RESISTANCE $f = 1\text{ kHz}$	
	V _Z at I _{ZT1}	I _{ZT1}	I _{ZT2}	I _R at V _R	Z _Z at I _{ZT1} (I ^t)	Z _{ZK} at I _{ZT2}	α_{VZ}
	V	mA		μA	V	Ω	%/K
NOM.				MAX.		MAX.	TYP.
1N5221B	2.4	20	0.25	100	1	30	1200
1N5222B	2.5	20	0.25	100	1	30	1250
1N5223B	2.7	20	0.25	75	1	30	1300
1N5224B	2.8	20	0.25	75	1	30	1400
1N5225B	3	20	0.25	50	1	29	1600
1N5226B	3.3	20	0.25	25	1	28	1600
1N5227B	3.6	20	0.25	15	1	24	1700
1N5228B	3.9	20	0.25	10	1	23	1900

La distorsión de la señal comienza aproximadamente a los 3.12 V. Para la limitación de tensiones negativas se colocó el mismo diodo 10BQ015.

La elección de este diodo y no del 1N5225B se debe a que este último comienza a distorsionar la señal de entrada en aproximadamente 2.8 V, lo que limita aún más el rango de entrada. Por otro lado, que la señal de entrada supere levemente el rango de entrada no es significativo, ya que la hoja de especificaciones del conversor ADS1256 toma ciertos márgenes de seguridad y se realizaron una gran cantidad de pruebas para cerciorarse de que las entradas no sufren daños al ingresar tensiones de 3.3 V y pequeñas tensiones

negativas. De igual manera se recomienda mantener los niveles de tensión de las señales de entrada dentro del rango especificado.

Las simulaciones realizadas arrojaron lo siguiente:

- Sin superar los 3 V:

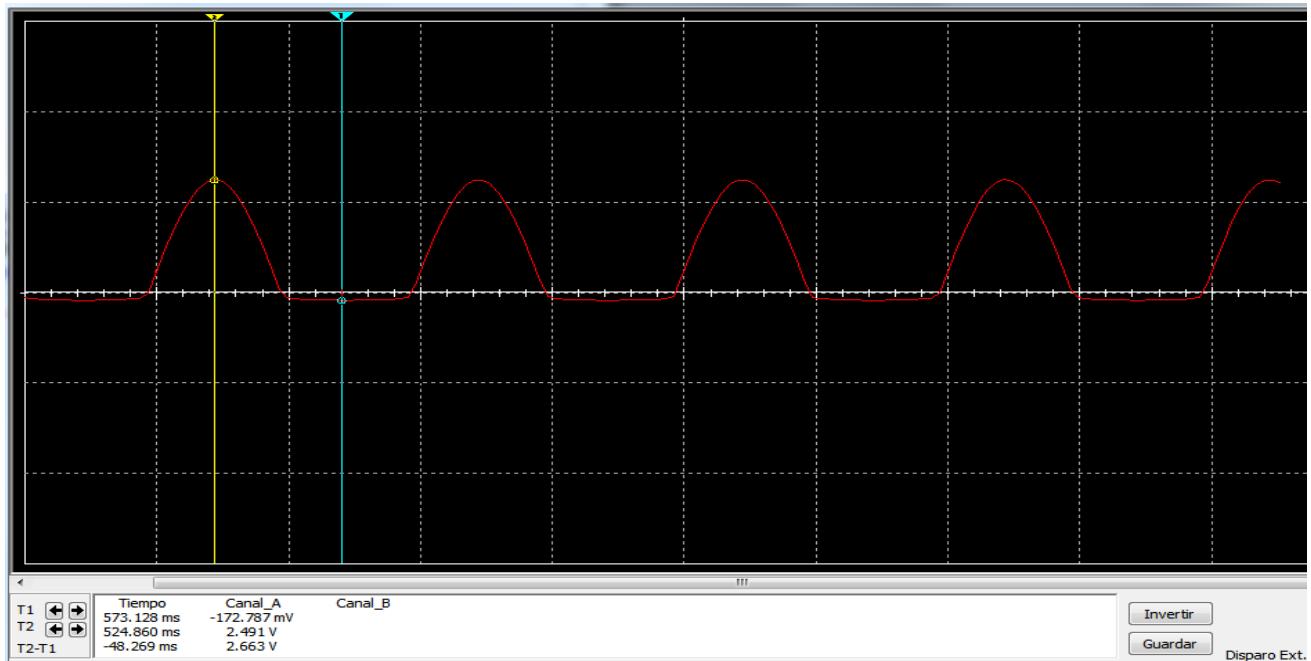
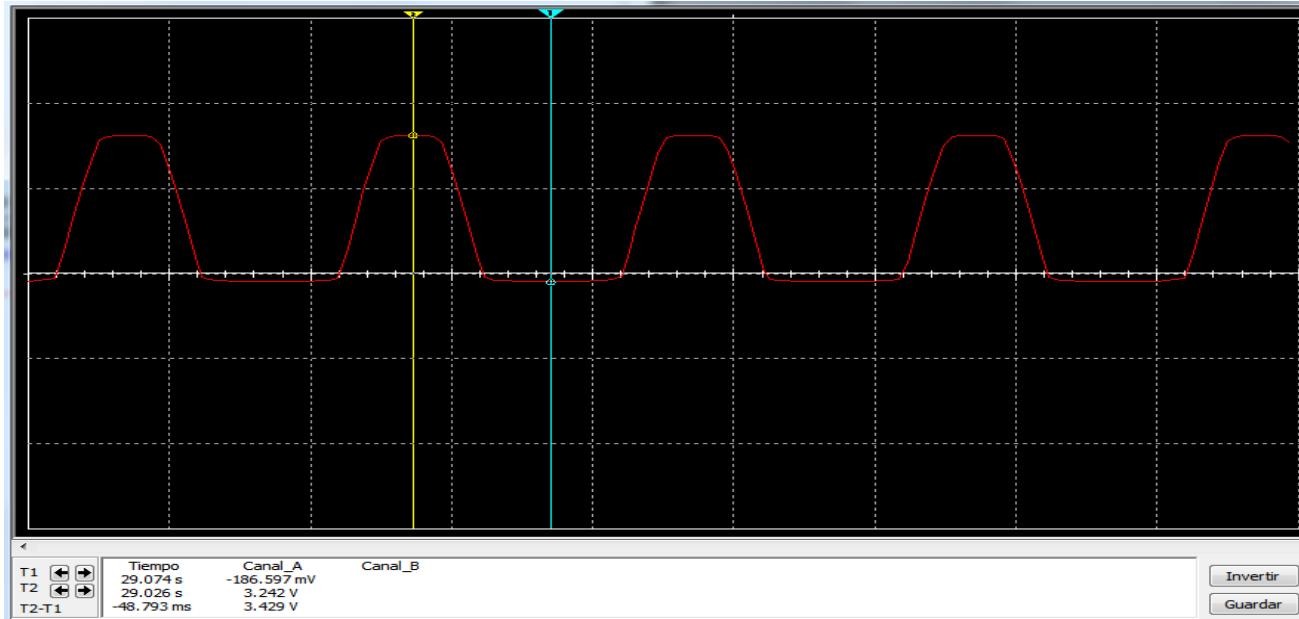


Figura 13: Simulación de la señal obtenida sin superación del rango máximo con buffer habilitado.

Puede observarse que la señal de entrada no sufre distorsión en el semicírculo positivo al ingresar una señal de 2.5 V. Por el contrario, el semicírculo negativo se ve recortado a los -0.172 V.

➤ Superando los 3 V:



En cambio, al inyectar una señal de entrada de 4 V pico, el semiciclo positivo es recortado a los 3.24 V.

Una vista de este circuito puede verse a continuación:



Figura 15: Vista superior de la placa PCB del circuito de protección.

5.1.4 Módulo RTC:

Como se pudo comprobar en base a pruebas, la placa Raspberry Pi no posee reloj interno en tiempo real. Esto trae el inconveniente de que, si la misma se inicia sin conexión a internet, utilizará la fecha y hora de la última vez que fue iniciada y no con la actual.

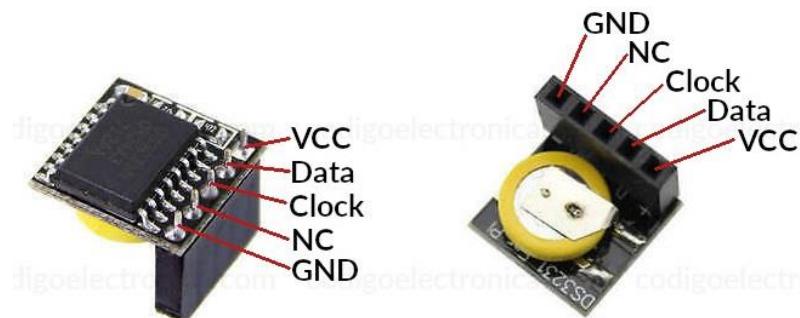
Es por esto que debemos recurrir a un módulo RTC que posea una pequeña pila y que permita mantener la hora y la fecha aún si la placa no está alimentada.

Se evaluaron diversas alternativas, como el RTC DS1307, el DS1302 y el DS3231. Por sus características, precio, cantidad de información disponible y comodidad de uso, se optó por la última alternativa, el DS3231.

Sus principales características son:

- Voltaje de Alimentación DC: 3.3V ~ 5V
- Bajo consumo de energía
- Tipo de Comunicación: I2C
- Soporta el Calendario hasta el año 2100
- Compensación de año bisiesto.
- Formato de hora configurable en 12 o 24 horas
- 2 alarmas configurables
- Frecuencia de Funcionamiento: 400 KHz
- Frecuencia de Salida: 1 Hz y 32.768 KHz
- Temperatura de Funcionamiento: -40 °C ~ 85 °C
- Precisión del sensor de temperatura interno: ± 3 °C
- Dimensiones: 13 mm x 13 mm x 14 mm

Configuración de pines y conexión a la placa Raspberry Pi:



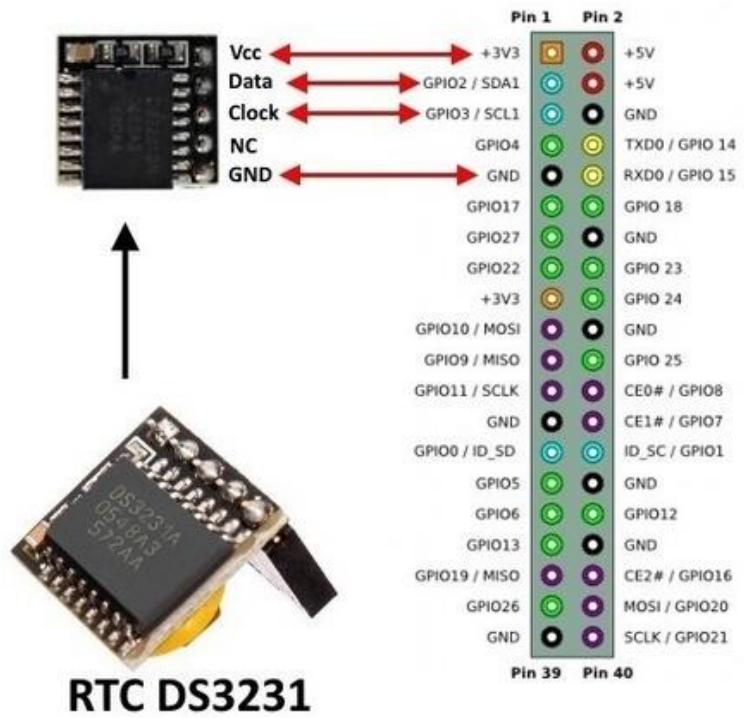


Figura 16: Configuración de pines del módulo RTC.

El montaje de este módulo en la placa de la Raspberry Pi puede observarse en la siguiente imagen:



Figura 17: Conexión del módulo RTC a la Raspberry Pi.

Precaución: la conexión del módulo a la Raspberry Pi debe hacerse con esta última apagada.

5.1.5 Esquema de conexión completo:

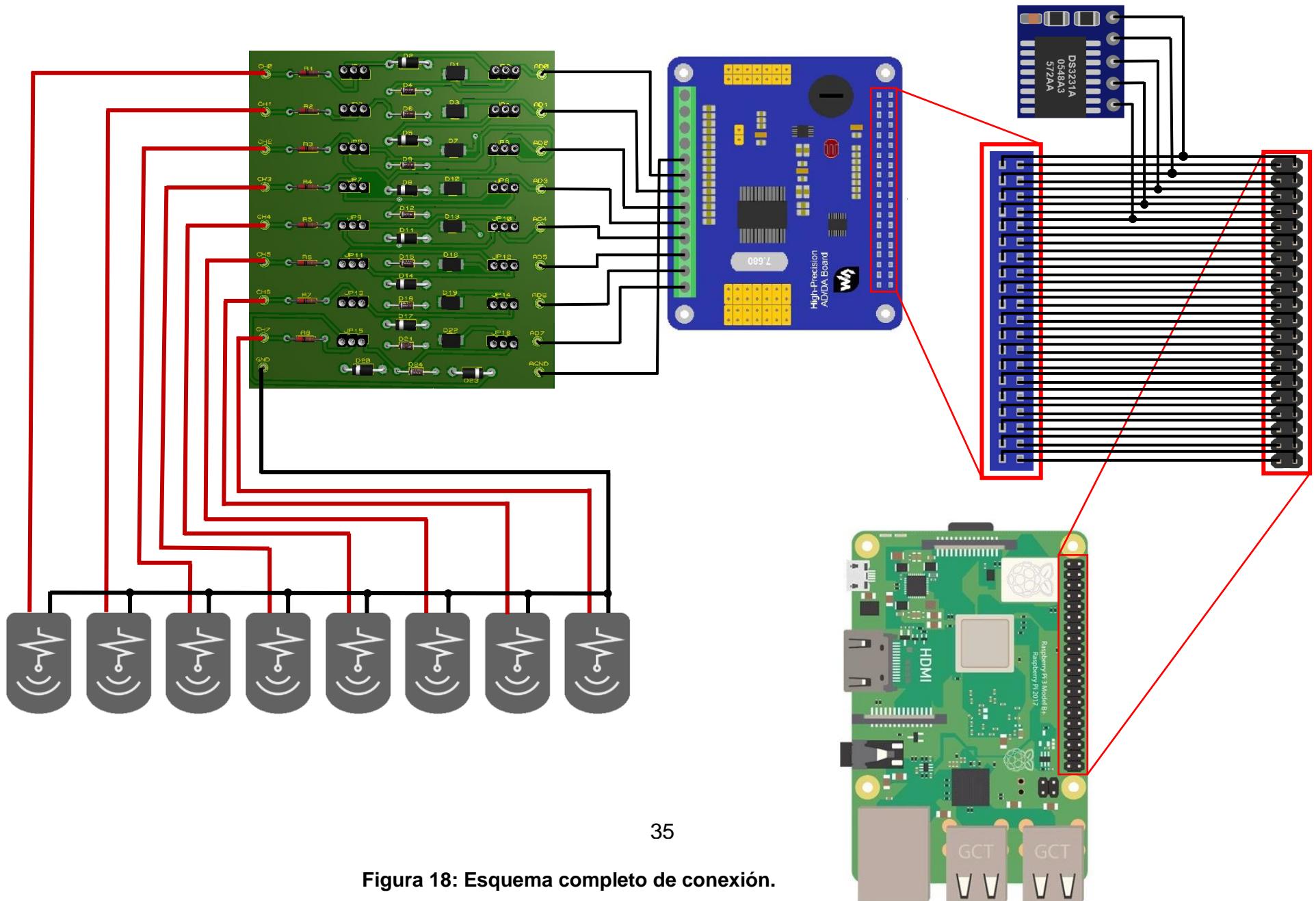


Figura 18: Esquema completo de conexión.

El montaje entre la placa Raspberry Pi y la placa conversora puede observarse en la figura siguiente:

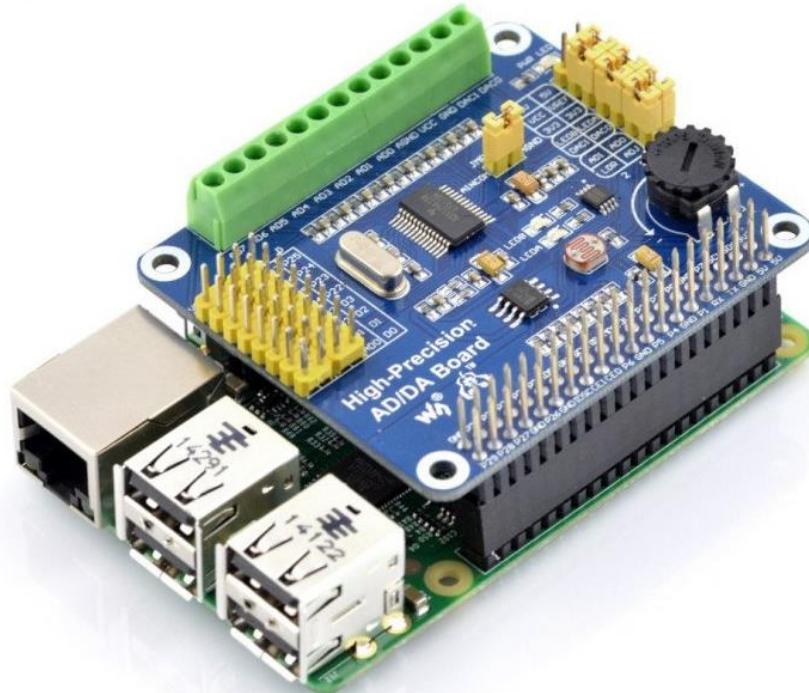
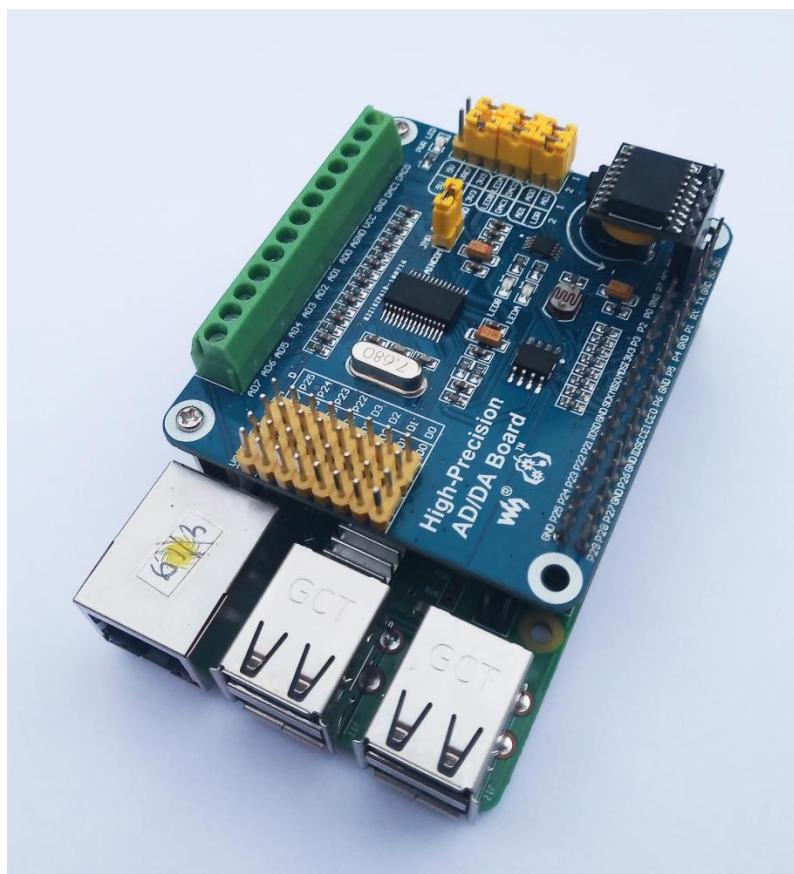


Figura 19: Montaje de la placa conversora sobre la Raspberry Pi.

Y el montaje final, con el módulo RTC sin el circuito de protección, es el siguiente:





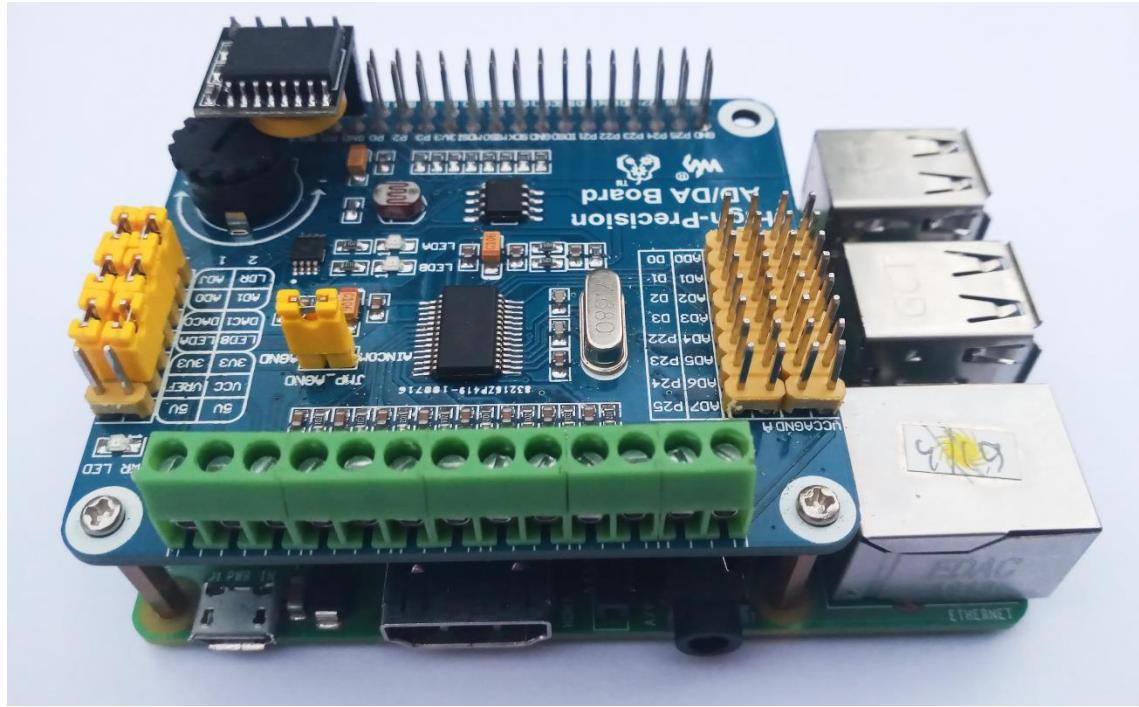


Figura 20: Montaje final del dispositivo.

5.2 SOFTWARE

Las librerías y funciones empleadas en el programa encargado de realizar las mediciones fueron obtenidas a través de la página <https://github.com/AKEOPLUS-boris-bocquet/RaspberryPi-ADC-DAC>. Entre ellas se encuentra la biblioteca BCM2835, necesaria para poder manejar las salidas de la placa, ya que proporciona acceso a las GPIO (General Purpose Input/Output) y otras funciones de E/S basadas en el chip Broadcom BCM 2835, utilizado por la placa Raspberry Pi, permitiendo a la misma el acceso a los pines GPIO en el conector IDE de 26 pines, necesarios para controlar e interactuar con varios dispositivos externos.

Se siguieron los pasos mencionados en la página antes citada. En la misma se descargan las librerías y códigos para realizar la conversión analógica a digital (ADC, usando el circuito ADS1256) y digital a analógico (DAC, usando DAC8552) basados en Raspberry PI extendido con Waveshare High-Precision AD-DA Raspberry Pi hat/shield (placa conversora).

El primer paso es descargar la biblioteca BCM2835 y los encabezados. Descargar la última versión de la biblioteca bcm2835-1.xx.tar.gz ejecutando los siguientes comandos:

```
tar zxvf bcm2835-1.xx.tar.gz
cd bcm2835-1.xx
./configure
make
sudo make check
sudo make install
```

El segundo paso es descargar e instalar el compilador cmake, así como también software de control de versiones Git con el fin de descargar estas librerías desde GitHub.

Aplicar los comandos:

```
sudo apt-get update  
sudo apt-get install git cmake.
```

Luego puede clonarse el repositorio a la placa Raspberry Pi:

```
git clone https://github.com/AKEOPLUS-boris-bocquet/RaspberryPi-ADC-DAC.git  
cd RaspberryPi-ADC-DAC  
mkdir build  
cd build  
cmake ..  
make
```

Antes de poder compilar y ejecutar cualquier programa se debe habilitar la comunicación SPI. Por defecto esta comunicación viene deshabilitada. Existen dos formas de habilitarla:

1. Desde la terminal:

Aplicar `sudo raspi-config` con lo cual accederá a las configuraciones de la Raspberry Pi y luego ir a *5 – Interfacing Options* y *P4 - SPI* y colocar *<Yes>* para habilitar SPI.

2. Desde el GUI de la Raspberry Pi dirigirse a las configuraciones, luego a opciones de interfaz y entonces marcar el casillero de habilitar en la comunicación SPI.

Si se ha instalado todo correctamente, compile el programa `mainTestingAdda.c` con `sudo make mainTestingAdda.c`. Deberá crearse el archivo ejecutable `testAdda`. Coloque los puentes de la placa conversora de la siguiente manera:

- VCC a 5V
- VREF a 5V
- DAC0 a LEDA
- DAC1 a LEDB
- AD0 a ADJ
- AD1 a LDR

Esto se hace con jumpers en la placa en pines referenciados. Puede colocar VCC y VREF a los 3.3 V en vez de los 5 V, debido a que esta tensión de referencia es utilizada esencialmente por el DAC. Esta configuración implica que la salida DA0 se conecta al LED A, el cual se encuentra incluido en la placa, y la salida DA1 al LED B. Por otro lado, la entrada

AD0 se conecta a ADJ, el cual hace referencia al potenciómetro que viene incluido en la misma placa y AD1 se conecta al LDR, por lo que, modificando la posición del potenciómetro y la luz incidente sobre el LDR, modificamos los niveles de luz emitida por los leds A y B.

Para ejecutar el programa mainTestingAdda.c, que es el ejemplo de prueba realizamos:

```
sudo ./testAdda
```

Se mostrarán por consola las sucesivas mediciones que se realizan en cada uno de los 8 canales del ADC.

Notar que la ejecución de los programas se realiza con sudo, ya que se requiere permisos de root o super usuario para poder ejecutarlas.

Las mediciones se mostrarán en voltios, estando estas entradas al aire sin conectar. Esto se debe a que dichas entradas analógicas poseen una muy alta impedancia de entrada, por lo que cualquier ruido por más pequeño que sea, causará un acuse de medición. Esto se debe tener en cuenta a la hora de realizar futuras mediciones, por lo que el recinto donde se encontrará la placa debería, en lo posible, estar idealmente blindado para evitar interferencias.

Otra posible solución es medir en forma diferencial para anular los ruidos que ingresen a ambas entradas, con la desventaja de que solo se podrá utilizar la mitad de los canales, es decir 4 de los 8 que posee el conversor.

También se debe tener en cuenta que, en caso de conectar las entradas a un potencial nulo, las mediciones arrojarán entre 0.001 y 0.05 V, ya que los mismos cables de conexión se ven inducidos con pequeñas interferencias. Es, por lo tanto, muy difícil lograr un potencial realmente nulo.

5.2.1 REGISTROS

Se presentarán los registros más importantes a tener en cuenta para el desarrollo posterior del programa de medición.

Mapa de registros:

ADDRESS	REGISTER	RESET VALUE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
00h	STATUS	x1H	ID3	ID2	ID1	ID0	ORDER	ACAL	BUFEN	DRDY
01h	MUX	01H	PSEL3	PSEL2	PSEL1	PSEL0	NSEL3	NSEL2	NSEL1	NSEL0
02h	ADCON	20H	0	CLK1	CLK0	SDCS1	SDCS0	PGA2	PGA1	PGA0
03h	DRATE	F0H	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
04h	IO	E0H	DIR8	DIR2	DIR1	DIR0	DIO3	DIO2	DIO1	DIO0
05h	OFC0	xxH	OFC07	OFC06	OFC05	OFC04	OFC03	OFC02	OFC01	OFC00
06h	OFC1	xxH	OFC15	OFC14	OFC13	OFC12	OFC11	OFC10	OFC09	OFC08
07h	OFC2	xxH	OFC23	OFC22	OFC21	OFC20	OFC19	OFC18	OFC17	OFC16
08h	FSC0	xxH	FSC07	FSC06	FSC05	FSC04	FSC03	FSC02	FSC01	FSC00
09h	FSC1	xxH	FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC09	FSC08
0Ah	FSC2	xxH	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16

1. Registro STATUS:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID	ID	ID	ID	ORDER	ACAL	BUFEN	DRDY

En la librería AD-DA-WS-RPI.h tiene su dirección igualada a 0. Los primeros 4 bits más significativos son de identificación y no se utilizan. El bit 3 especifica el orden en que saldrán los datos; si toma el valor 0 saldrán primero del bit más significativo al menos significativo. Si su valor es 1 será el caso contrario.

El bit 2 habilita la auto calibración: en 0 se encuentra deshabilitada y en 1 habilitada. Si es habilitada, esta auto calibración comenzará luego de modificar el PGA en el registro ADCON, el DR en el registro DRATE o BUFEN en el registro STATUS.

El bit 1 (BUFEN) tiene como función habilitar o deshabilitar el buffer analógico de entrada. Con esto se consigue una mayor o menor impedancia de entrada ya que, cuando está deshabilitada, la impedancia diferencial depende del PGA y su valor se encuentra entre los 4.7 kΩ a los 150/PGA Ω, mientras que cuando está habilitada llega a 80 MΩ mientras la frecuencia de los datos sea menor a los 50 Hz. A través de las pruebas realizadas pudimos determinar que, aún con el buffer deshabilitado, las impedancias que presentan las entradas analógicas son superiores a 1 MΩ. En 0 esta deshabilitado el buffer y en 1 habilitado.

Bit 0 (DRDY*): este bit duplica el estado del pin DRDY, terminal que indica cuando una conversión ha finalizado y los datos se encuentran listos para ser leídos. Este pin toma un estado alto cuando la conversión comienza y un estado bajo cuando esta termina. Por esto, se utilizará este pin para determinar cuando los datos estén listos o se leerá el valor de este bit 0 que refleja el mismo estado.

2. Registro MUX:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
PSEL3	PSEL2	PSEL1	PSEL0	NSEL3	NSEL2	NSEL1	NSEL0

Este registro nos permite conmutar los distintos canales de entrada del conversor. Del bit 7 al 4 permite conmutar los 8 canales con valores positivos de tensión. Del 3 al 0 lo mismo, pero con valores de tensión de referencia.

Bits 7-4 **PSEL3, PSEL2, PSEL1, PSEL0**: Positive Input Channel (AIN_P) Select

0000 = AIN_0 (default)
0001 = AIN_1
0010 = AIN_2 (ADS1256 only)
0011 = AIN_3 (ADS1256 only)
0100 = AIN_4 (ADS1256 only)
0101 = AIN_5 (ADS1256 only)
0110 = AIN_6 (ADS1256 only)
0111 = AIN_7 (ADS1256 only)

1xxx = AIN_{COM} (when PSEL3 = 1, PSEL2, PSEL1, PSEL0 are “don’t care”)

NOTE: When using an ADS1255 make sure to only select the available inputs.

Bits 3-0 **NSEL3, NSEL2, NSEL1, NSEL0**: Negative Input Channel (AIN_N)Select

0000 = AIN_0
0001 = AIN_1 (default)
0010 = AIN_2 (ADS1256 only)
0011 = AIN_3 (ADS1256 only)
0100 = AIN_4 (ADS1256 only)
0101 = AIN_5 (ADS1256 only)
0110 = AIN_6 (ADS1256 only)
0111 = AIN_7 (ADS1256 only)

1xxx = AIN_{COM} (when NSEL3 = 1, NSEL2, NSEL1, NSEL0 are “don’t care”)

Los pasos para cambiar de canal y leer son los siguientes:

1_ Cuando el pin DRDY pasa a estado bajo, indicando que la conversión finalizó y los datos se encuentran disponibles, actualizar el registro MUX para cambiar la entrada del multiplexor y por lo tanto el canal a medir, usando el comando WREG. Por ejemplo, establecer MUX en 23h da $AIN_P = AIN_2$, $AIN_N = AIN_3$. En nuestro caso se debería usar el terminal AIN_{COM} como referencia.

2_Reinic peaceo de conversión emitiendo un comando SYNC seguido inmediatamente por un comando WAKEUP. Asegúrese de seguir la especificación de tiempo t11 entre los comandos.

3_Lea los datos de la conversión anterior usando el comando RDATA.

4_Cuando DRDY vuelve a estado bajo, repita el ciclo actualizando primero el registro del multiplexor y luego leyendo los datos anteriores.

Cuando se conmuta de canales, trae como consecuencia un retraso adicional que produce que la frecuencia máxima de muestreo efectiva de cada canal disminuya:

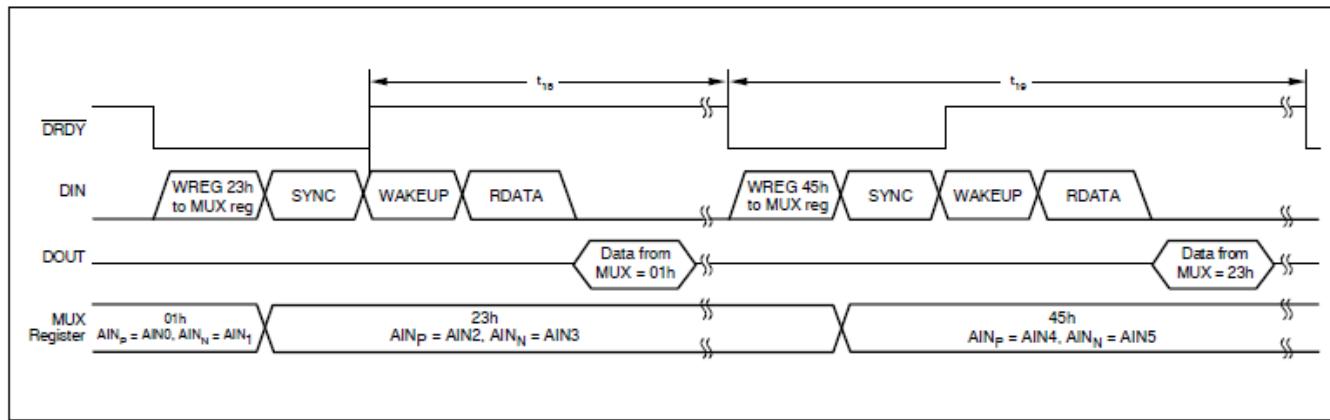


Figura 21: Ciclo del multiplexor de entrada del ADS1256.

Y las frecuencias máximas decaen a:

Table 14. Multiplexer Cycling Throughput

DATA RATE (SPS)	CYCLING THROUGHPUT ($1/t_{19}$) (Hz)
30,000	4374
15,000	3817
7500	3043
3750	2165
2000	1438
1000	837
500	456
100	98
60	59
50	50
30	30
25	25
15	15
10	10
5	5
2.5	2.5

NOTE: $f_{CLKIN} = 7.68\text{MHz}$.

3. Registro ADCON:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	CLK1	CLK0	SDCS1	SDCS0	PGA2	PGA1	PGA0

Los bits 6 y 5 permiten cambiar la frecuencia de salida del reloj. Esta ya viene predeterminada de forma tal que puedan obtenerse las 30000 muestras por segundo partiendo de fclkin (frecuencia del clock de en entrada) sin divisiones.

Los bits 4 y 3 sirven para la detección de niveles de corrientes provenientes de sensores externos

Bit 7 Reserved, always 0 (Read Only)

Bits 6-5 **CLK1, CLK0**: D0/CLKOUT Clock Out Rate Setting

00 = Clock Out OFF

01 = Clock Out Frequency = f_{CLKIN} (default)

10 = Clock Out Frequency = $f_{CLKIN}/2$

11 = Clock Out Frequency = $f_{CLKIN}/4$

When not using CLKOUT, it is recommended that it be turned off. These bits can only be reset using the **RESET** pin.

Bits 4-2 **SDCS1, SDCS0**: Sensor Detect Current Sources

00 = Sensor Detect OFF (default)

01 = Sensor Detect Current = $0.5\mu A$

10 = Sensor Detect Current = $2\mu A$

11 = Sensor Detect Current = $10\mu A$

The Sensor Detect Current Sources can be activated to verify the integrity of an external sensor supplying a signal to the ADS1255/6. A shorted sensor produces a very small signal while an open-circuit sensor produces a very large signal.

Bits 2-0 **PGA2, PGA1, PGA0**: Programmable Gain Amplifier Setting

000 = 1 (default)

001 = 2

010 = 4

011 = 8

100 = 16

101 = 32

110 = 64

111 = 64

Los bits 2,1 y 0 sirven para la configuración de la ganancia del amplificador de ganancia controlable (PGA) en la entrada. Permite obtener distintas escalas y sensibilidades del conversor.

Precaución: la ganancia del PGA afecta a una gran cantidad de parámetros, como la tensión de referencia, el ruido efectivo, la cantidad de bits libre de ruido, la frecuencia de salida de los datos ya que para distintas ganancias se intercalan distintas capacidades que afectan la respuesta en frecuencia, etc.

Un parámetro importante es la cantidad de bits libres de ruido, ya que son los bits que efectivamente poseen información relevante en las mediciones tomadas y es función de la ganancia:

DATA RATE (SPS)	PGA						
	1	2	4	8	16	32	64
2.5	23.0	22.6	22.1	21.7	21.3	20.8	19.7
5	22.3	22.4	21.9	21.3	20.7	20.3	19.3
10	22.3	22.0	21.6	21.0	20.4	19.9	18.9
15	22.0	21.7	21.3	20.7	20.1	19.3	18.7
25	21.7	21.4	21.1	20.5	19.7	19.2	18.5
30	21.8	21.3	20.8	20.4	19.8	19.0	18.1
50	21.3	21.1	20.4	19.9	19.4	18.8	17.9
60	21.3	20.9	20.5	19.8	19.3	18.8	17.8
100	20.9	20.7	20.2	19.6	19.1	18.5	17.4
500	20.1	19.6	19.1	18.6	18.0	17.3	16.3
1000	19.0	18.6	18.1	17.5	17.2	16.5	15.6
2000	18.5	18.1	17.8	17.0	16.6	16.1	15.3
3750	18.1	17.8	17.3	16.6	16.2	15.7	14.7
7500	17.7	17.3	16.9	16.2	15.8	15.3	14.4
15,000	17.3	17.0	16.5	15.9	15.5	14.9	13.9
30,000	17.1	16.7	16.4	15.9	15.4	14.6	13.8

*Bits libres de ruido en función de la ganancia del PGA y de la frecuencia de conversión, con buffer habilitado.

Puede observarse que a medida que aumenta la frecuencia de conversión y la ganancia, disminuyen los bits libres de ruido.

También se muestran los niveles de ruido referidos a la entrada en volts, los cuales aumentan con la frecuencia, pero disminuyen con la ganancia:

DATA RATE (SPS)	PGA						
	1	2	4	8	16	32	64
2.5	0.247	0.156	0.080	0.056	0.043	0.037	0.033
5	0.301	0.175	0.102	0.076	0.061	0.045	0.044
10	0.339	0.214	0.138	0.106	0.082	0.061	0.061
15	0.401	0.264	0.169	0.126	0.107	0.085	0.073
25	0.494	0.305	0.224	0.149	0.134	0.102	0.093
30	0.533	0.335	0.245	0.176	0.138	0.104	0.106
50	0.629	0.393	0.292	0.216	0.168	0.136	0.122
60	0.692	0.438	0.321	0.233	0.184	0.146	0.131
100	0.875	0.589	0.409	0.305	0.229	0.170	0.169
500	1.946	1.250	0.630	0.648	0.497	0.390	0.367
1000	2.931	1.891	1.325	1.070	0.689	0.512	0.486
2000	4.173	2.589	1.827	1.492	0.943	0.692	0.654
3750	5.394	3.460	2.376	1.865	1.224	0.912	0.906
7500	7.249	4.593	3.149	2.436	1.691	1.234	1.187
15,000	9.074	5.921	3.961	2.984	2.125	1.517	1.515
30,000	10.728	6.705	4.446	3.280	2.416	1.785	1.742

*Niveles de ruido de entrada (uV, rms), con buffer habilitado.

Y, por último, se indica el número efectivo de bits:

DATA RATE (SPS)	PGA						
	1	2	4	8	16	32	64
2.5	25.3	24.9	24.9	24.4	23.8	23.0	22.2
5	25.0	24.8	24.5	24.0	23.3	22.7	21.8
10	24.8	24.5	24.1	23.5	22.9	22.3	21.3
15	24.6	24.2	23.8	23.2	22.5	21.8	21.0
25	24.3	24.0	23.4	23.0	22.2	21.5	20.7
30	24.2	23.8	23.3	22.8	22.1	21.5	20.5
50	23.9	23.6	23.0	22.5	21.8	21.1	20.3
60	23.8	23.4	22.9	22.4	21.7	21.0	20.2
100	23.4	23.0	22.5	22.0	21.4	20.8	19.8
500	22.3	21.9	21.5	20.9	20.3	19.6	18.7
1000	21.7	21.3	20.8	20.2	19.8	19.2	18.3
2000	21.2	20.9	20.4	19.7	19.3	18.8	17.9
3750	20.8	20.5	20.0	19.4	19.0	18.4	17.4
7500	20.4	20.1	19.6	19.0	18.5	17.9	17.0
15,000	20.1	19.7	19.3	18.7	18.2	17.7	16.7
30,000	19.8	19.5	19.1	18.5	18.0	17.4	16.5

*Número efectivo de bits (ENOB, rms), con buffer habilitado.

El número efectivo de bits es una de las especificaciones dinámicas más utilizadas para convertidores A/D reales, ya que es un indicador global de la resolución del sistema para una determinada frecuencia de la señal de entrada a una cierta velocidad de muestreo. Se calcula en base a los datos digitales de salida del convertidor como:

$$ENOB = N - \log_2 \left(\frac{A_{\text{error_medido[rms]}}}{A_{\text{error_ideal[rms]}}} \right)$$

Donde N es la resolución del convertidor, es decir, el número de bits del mismo, Aerror_medido [rms] es el valor rms medio del ruido medido y Aerror_ideal [rms] representa el error de cuantización que puede ser expresado como:

$$A_{\text{error_ideal[rms]}} = \frac{LSB}{\sqrt{12}} = \frac{FS}{2^N \sqrt{12}} \quad LSB = \frac{FS}{2^N}$$

Dónde FS (*Full Scale*) representa el rango completo de valores analógicos de entrada y N la resolución del convertidor.

En resumen, se puede definir el ENOB como un método de especificación de la resolución del sistema e indica que el convertidor es equivalente a un ADC ideal de ese número (ENOB) de bits.

A pesar de ser una especificación dinámica, en ella quedan reflejadas todas las imperfecciones del sistema, es decir, engloba la totalidad de fuentes de ruido, siendo así la mejor manera de comprobar el rendimiento del convertidor.

Todas las tablas mostradas parten de que el buffer de entrada se encuentra habilitado, también están las tablas para la condición de buffer deshabilitado (véase anexo Hoja de datos del ADS1256).

Por otro lado, a medida que aumenta la ganancia aumenta la resolución:

PGA SETTING	FULL-SCALE INPUT VOLTAGE $V_{IN}^{(1)}$ ($V_{REF} = 2.5V$)
1	$\pm 5V$
2	$\pm 2.5V$
4	$\pm 1.25V$
8	$\pm 0.625V$
16	$\pm 312.5mV$
32	$\pm 156.25mV$
64	$\pm 78.125mV$

*Escala completa del voltaje de entrada vs ganancia del PGA.

La escala de voltaje de entrada es: $\pm 2V_{REF}/\text{PGA}$.

Por ejemplo, si la máxima señal que se va a medir es de 1V conviene usar la escala de $\pm 1.25V$, por lo que la PGA deberá establecerse en 4.

Tenga en cuenta que las escalas de tensión mostradas en la tabla anterior utilizan una tensión de referencia de 2.5 V.

También afecta, como se mencionó anteriormente, a la frecuencia de muestreo, ya que el tsample, tiempo que demora en realizar un ciclo completo del estado alto al estado bajo y nuevamente al estado alto, es función de la ganancia.

La estructura simplificada de la entrada del conversor con el buffer deshabilitado puede observarse en la siguiente figura:

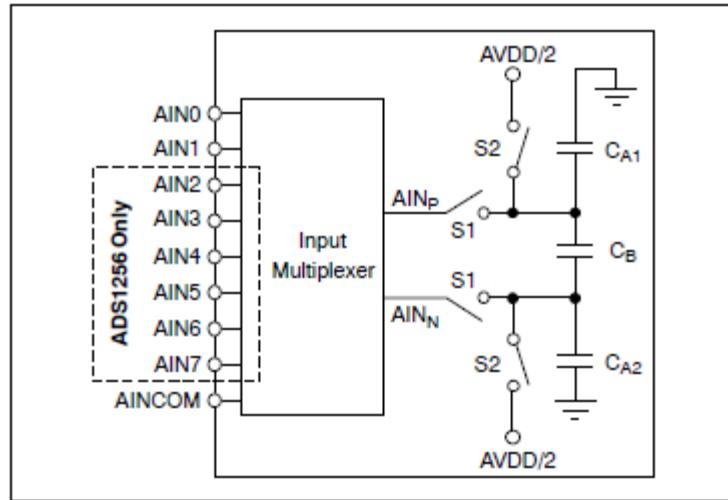


Figura 22: Estructura de entrada simplificada con buffer deshabilitado.

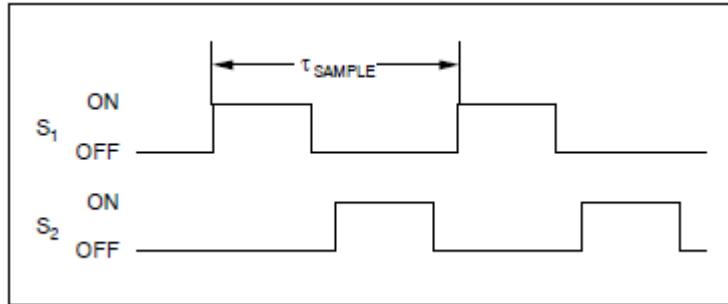


Figura 23: Tiempos de conmutación de los interruptores S1 y S2.

PGA SETTING	$\tau_{\text{SAMPLE}}^{(1)}$	C_A	C_B
1	$f_{\text{CLKIN}}/4$ (521ns)	2.1pF	2.4pF
2	$f_{\text{CLKIN}}/4$ (521ns)	4.2pF	4.9pF
4	$f_{\text{CLKIN}}/4$ (521ns)	8.3pF	9.7pF
8	$f_{\text{CLKIN}}/4$ (521ns)	17pF	19pF
16	$f_{\text{CLKIN}}/4$ (521ns)	33pF	39pF
32	$f_{\text{CLKIN}}/2$ (260ns)	33pF	39pF
64	$f_{\text{CLKIN}}/2$ (260ns)	33pF	39pF

(1) τ_{SAMPLE} for $f_{\text{CLKIN}} = 7.68\text{MHz}$.

*Tiempo de muestreo τ_{SAMPLE} , y CA y CB vs la ganancia del PGA.

Por último, la impedancia de entrada con el buffer deshabilitado es:

PGA SETTING	Z_{effA} (kΩ)	Z_{effB} (kΩ)
1	260	220
2	130	110
4	65	55
8	33	28
16	16	14
32	8	7
64	8	7

*Impedancias analógicas efectivas de entrada con buffer deshabilitado.

4. Registro DRATE:

Registro que permite configurar la frecuencia de muestreo. Dichas frecuencias se obtienen para fclkin= 7.68 MHz y sin conmutación de los canales de entrada.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

Bits 7-0 DR[7: 0]: Data Rate Setting

11110000 = 30,000SPS (default)
 11100000 = 15,000SPS
 11010000 = 7,500SPS
 11000000 = 3,750SPS
 10110000 = 2,000SPS
 10100001 = 1,000SPS
 10010010 = 500SPS
 10000010 = 100SPS
 01110010 = 60SPS
 01100011 = 50SPS
 01010011 = 30SPS
 01000011 = 25SPS
 00110011 = 15SPS
 00100011 = 10SPS
 00010011 = 5SPS
 00000011 = 2.5SPS

5. Registro I/O GPIO:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DIR3	DIR2	DIR1	DIR0	DIO3	DIO2	DIO1	DIO0

Posee 4 pines de entrada/salida: D3, D2, D1 y D0/CLKOUT.

- Bit 7 **DIR3**, Digital I/O Direction for Digital I/O Pin D3 (used on ADS1256 only)
 0 = D3 is an output
 1 = D3 is an input (default)
 - Bit 6 **DIR2**, Digital I/O Direction for Digital I/O Pin D2 (used on ADS1256 only)
 0 = D2 is an output
 1 = D2 is an input (default)
 - Bit 5 **DIR1**, Digital I/O Direction for Digital I/O Pin D1
 0 = D1 is an output
 1 = D1 is an input (default)
 - Bit 4 **DIR0**, Digital I/O Direction for Digital I/O Pin D0/CLKOUT
 0 = D0/CLKOUT is an output (default)
 1 = D0/CLKOUT is an input
- Bits 3-0 DIO[3:0]: Status of Digital I/O Pins D3, D2, D1, D0/CLKOUT**

Los bits del 3 al 0 tienen como función que, si los bits superiores se configuran como salidas, al escribir en estos bits DI/O se obtendrá dicho valor a la salida en el pin correspondiente, el DIO3 con el DIR3, el DIO2 con el DIR2 y así sucesivamente. Si se configuran como entrada entonces escribir en los bits DI/O no tendrá ningún efecto al igual que tampoco se obtendrá el CLKOUT.

5.2.2. COMANDOS

Conocer los comandos principales es esencial para entender cómo funcionan las funciones implementadas en las librerías necesarias para el uso de la placa conversora. También el conocimiento de estos comandos servirá en caso de que se desee un control más preciso sobre el comportamiento y la operación del conversor. Es por esto por lo que se explicarán levemente aquellos comandos más utilizados. Un listado completo de los mismos puede observarse a continuación:

COMMAND	DESCRIPTION	1ST COMMAND BYTE	2ND COMMAND BYTE
WAKEUP	Completes SYNC and Exits Standby Mode	0000 0000 (00h)	
RDATA	Read Data	0000 0001 (01h)	
RDATAC	Read Data Continuously	0000 0011 (03h)	
SDATAC	Stop Read Data Continuously	0000 1111 (0Fh)	
RREG	Read from REG <i>rrr</i>	0001 <i>rrr</i> (1xh)	0000 <i>nnnn</i>
WREG	Write to REG <i>rrr</i>	0101 <i>rrr</i> (5xh)	0000 <i>nnnn</i>
SELF CAL	Offset and Gain Self-Calibration	1111 0000 (F0h)	
SEFOCAL	Offset Self-Calibration	1111 0001 (F1h)	
SELFGCAL	Gain Self-Calibration	1111 0010 (F2h)	
SYSOCAL	System Offset Calibration	1111 0011 (F3h)	
SYSGCAL	System Gain Calibration	1111 0100 (F4h)	
SYNC	Synchronize the A/D Conversion	1111 1100 (FCh)	
STANDBY	Begin Standby Mode	1111 1101 (FDh)	
RESET	Reset to Power-Up Values	1111 1110 (FEh)	
WAKEUP	Completes SYNC and Exits Standby Mode	1111 1111 (FFh)	

*n = número de registros a leer/escribir – 1. Por ejemplo, para leer/escribir 1 registro, configure nnnn = 2 (0010).

r = dirección inicial de registro para comandos de lectura/escritura.

- **RDATA: Read Data**

Emita este comando después de que DRDY pase a estado bajo para leer un único resultado de conversión. Después de que los 24 bits se hayan desplazado hacia fuera en DOUT, DRDY pasa a estado alto. No es necesario volver a leer los 24 bits, pero DRDY no volverá a estado alto hasta que se actualicen nuevos datos.

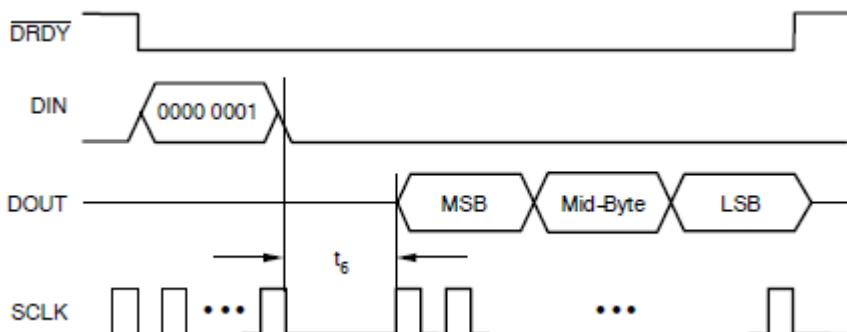


Figura 24: Secuencia del comando RDATA.

- **RDATAC: Read Data Continuous**

Emita el comando después de que DRDY pase a estado bajo para ingresar al modo continuo de lectura de datos. Este modo permite la salida continua de nuevos datos en cada DRDY sin la necesidad de emitir comandos de lectura posteriores.

Una vez leídos los 24 bits, DRDY pasa a alto. No es necesario volver a leer los 24 bits, pero DRDY no volverá alto hasta que se actualicen nuevos datos. Este modo puede terminarse con el comando detener lectura continua de datos (SDATAC).

Debido a que DIN se monitorea constantemente durante el modo de lectura continua de datos para el comando SDATAC o RESET, no use este modo si DIN y DOUT están conectados juntos.

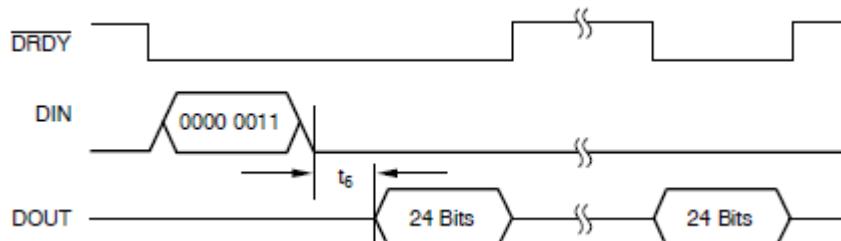


Figura 25: Secuencia del comando RDATAC.

Tener en cuenta que existe un retraso t6 (equivalente a 50 períodos del clock) entre que se emite el comando y efectivamente aparecen los datos en la salida.

RELOJ SERIE (SCLK)

El reloj en serie (SCLK) cuenta con una entrada activada por Schmitt y es utilizado para sincronizar los datos en los pines DIN y DOUT dentro y fuera del ADS1256. Aunque la entrada posee histéresis, se recomienda mantener SCLK lo más limpio posible para evitar que los fallos modifiquen accidentalmente los datos. Si SCLK se mantiene bajo durante 32 períodos DRDY, la interfaz en serie se reiniciará y el siguiente pulso SCLK iniciará un nuevo ciclo de comunicación. Esta función de tiempo de espera se puede utilizar para recuperar la comunicación cuando se interrumpe la transmisión de una interfaz en serie. Un patrón especial en SCLK reiniciará el chip. Cuando la interfaz serial esté inactiva, mantenga SCLK bajo.

- **SDATAC: Stop Read Data Continuous**

Finaliza el modo de salida de datos continua. El comando debe emitirse después de que DRDY baje y se debe completar antes de que DRDY suba.

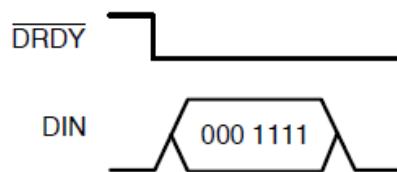


Figura 26: Secuencia del comando SDATAC.

- **RREG: Read from Registers**

Imprime los datos de hasta 11 registros comenzando con la dirección de registro especificada como parte del comando. El número de registros leídos será uno más el segundo byte del comando. Si el recuento supera los registros restantes, las direcciones volverán al principio.

1er byte de comando: 0001 rrrr, donde rrrr es la dirección del primer registro a leer.

2do byte de comando: 0000 nnnn, donde nnnn es el número de bytes a leer - 1.

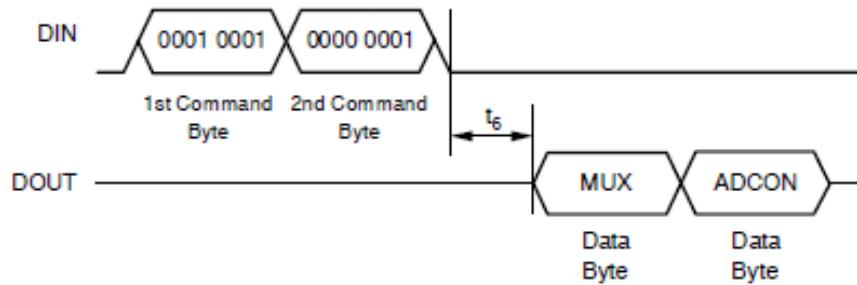


Figura 27: Secuencia del comando RREG.

- **WREG: Write to Register**

Escribe en los registros comenzando con el registro especificado como parte del comando. El número de registros que se escribirán es uno más el valor del segundo byte del comando.

1er byte de comando: 0101 rrrr, donde rrrr es la dirección del primer registro que se escribirá.

2do Byte de comando: 0000 nnnn, donde nnnn es el número de bytes que se escribirán - 1.

Byte (s) de datos: datos que se escribirán en los registros.

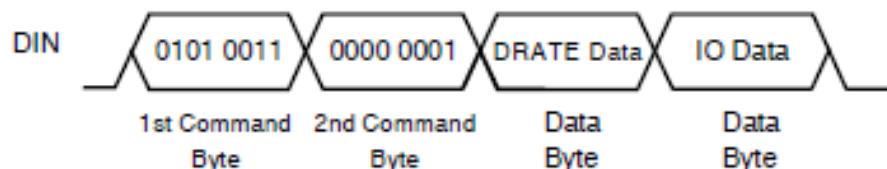


Figura 28: Secuencia del comando WREG.

Los comandos: **SELF CAL: Self Offset and Gain Calibration**, **SELF O FOCAL: Self Offset Calibration**, **SELF G CAL: Self Gain Calibration**, **SYS O CAL: System Offset Calibration** y **SYS G CAL: System Gain Calibration** son comandos para la auto/calibración del offset y la ganancia.

- **SYNC: Synchronize the A/D Conversion**

Este comando sincroniza la conversión A / D. La sincronización ocurre en el primer flanco ascendente CLKIN después del primer SCLK usado para ingresar el comando WAKEUP.

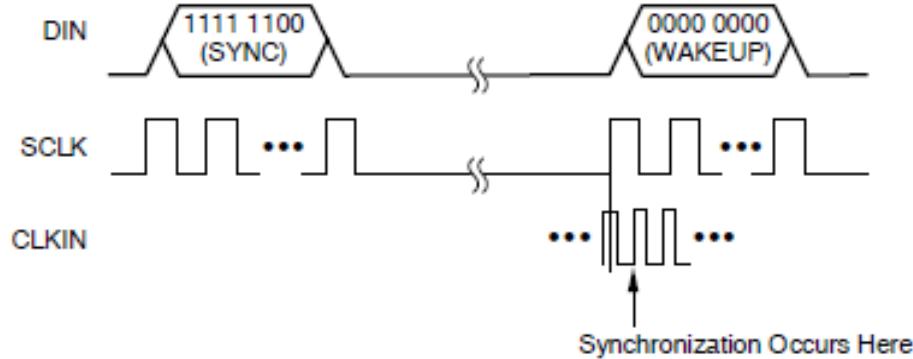


Figura 29: Secuencia del comando SYNC.

- **STANDBY: Standby Mode / One-Shot Mode**

Este comando pone el ADS1256 en un modo de espera de bajo consumo. Después de emitir el comando STANDBY, asegúrese de que no haya más actividad en SCLK mientras CS esté bajo, ya que esto interrumpirá el modo de espera. Si CS es alto, se permite la actividad de SCLK durante el modo de espera. Para salir del modo de espera, ejecute el comando WAKEUP. Este comando también se puede utilizar para realizar conversiones individuales.

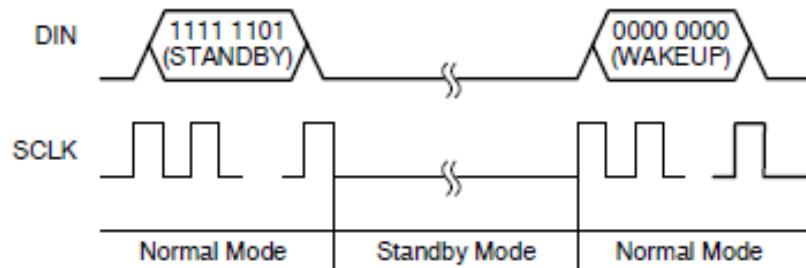


Figura 30: Secuencia del comando STANDBY.

- **WAKEUP: Complete Synchronization or Exit Standby Mode**

Se utiliza junto con los comandos SYNC y STANDBY. Hay dos valores (todos ceros o todos unos) disponibles para este comando.

- **RESET: Reset Registers to Default Values**

Devuelve todos los registros excepto los bits CLK0 y CLK1 en el registro ADCON a sus valores predeterminados.

Este comando también detendrá el modo de lectura continua: en este caso, emita el comando RESET después de que DRDY baje.

5.2.3 LIBRERÍAS E INSTRUCCIONES

La librería base es *AD-DA-WS-RPI.h* la cual llama a todas las demás librerías. El programa *AD-DA-WS-RPI.c* es el que implementa las instrucciones que utilizaremos.

Estas bibliotecas están basadas en el chip Broadcom BCM2835, circuito integrado encargado de las comunicaciones de la placa Raspberry Pi. Por lo tanto, permite que la placa Raspberry Pi pueda comunicarse con los puertos del conversor A/D.

En esta biblioteca hay variables definidas que usaremos.

Para la selección de la ganancia del PGA:

```
1 typedef enum
2 {
3     ADS1256_GAIN_1 = 0, /*GAIN 1 */
4     ADS1256_GAIN_2 = 1, /*GAIN 2 */
5     ADS1256_GAIN_4 = 2, /*GAIN 4 */
6     ADS1256_GAIN_8 = 3, /*GAIN 8 */
7     ADS1256_GAIN_16 = 4, /*GAIN 16 */
8     ADS1256_GAIN_32 = 5, /*GAIN 32 */
9     ADS1256_GAIN_64 = 6, /*GAIN 64 */
10 } ADS1256_GAIN_E;
```

Y para la frecuencia de muestreo:

```
1 typedef enum
2 {
3     ADS1256_30000SPS = 0,
4     ADS1256_15000SPS,
5     ADS1256_7500SPS,
6     ADS1256_3750SPS,
7     ADS1256_2000SPS,
8     ADS1256_1000SPS,
9     ADS1256_500SPS,
10    ADS1256_100SPS,
11    ADS1256_60SPS,
12    ADS1256_50SPS,
13    ADS1256_30SPS,
14    ADS1256_25SPS,
15    ADS1256_15SPS,
16    ADS1256_10SPS,
17    ADS1256_5SPS,
18    ADS1256_2d5SPS,
19    ADS1256_DRATE_MAX
20 } ADS1256_DRATE_E;
```

Para la elección de 8 entradas individuales o 4 entradas en modo diferencial:

```

1 typedef enum
2 {
3     SINGLE_ENDED_INPUTS_8 = (uint8_t)0,
4     DIFFERENTIAL_INPUTS_4 = (uint8_t)1,
5 } ADS1256_SCAN_MODE;

```

En modo single se toma cada entrada referida al terminal AINCOM, mientras que en modo diferencial se toma una entrada como positiva y la siguiente como referencia.

Para referenciar un registro en caso de requerir modificarlo o leer de él, se debe hacer referencia a la dirección de cada uno de ellos:

```

1 enum
2 {
3     /*Register address, followed by reset the default values */
4     REG_STATUS = 0, // x1H
5     REG_MUX = 1, // 01H
6     REG_ADCON = 2, // 20H
7     REG_DRATE = 3, // F0H
8     REG_IO = 4, // E0H
9     REG_OFC0 = 5, // xxH
10    REG_OFC1 = 6, // xxH
11    REG_OFC2 = 7, // xxH
12    REG_FSC0 = 8, // xxH
13    REG_FSC1 = 9, // xxH
14    REG_FSC2 = 10, // xxH
15 };

```

Estas direcciones coinciden con las indicadas en la hoja de datos del ADS1256. Por ejemplo, el registro STATUS se encuentra en la dirección 0h, el MUX en la 01h y así sucesivamente.

Para invocar un comando se debe ingresar por el terminal DIN las siguientes secuencias de bits (esto lo harán las instrucciones implementadas):

```

1 enum
2 {
3     CMD_WAKEUP = 0x00, // Completes SYNC and Exits Standby Mode 0000 0000 (00h)
4     CMD_RDATA = 0x01, // Read Data 0000 0001 (01h)
5     CMD_RDATAC = 0x03, // Read Data Continuously 0000 0011 (03h)
6     CMD_SDATAC = 0x0F, // Stop Read Data Continuously 0000 1111 (0Fh)
7     CMD_RREG = 0x10, // Read from REG rrr 0001 rrrr (1xh)
8     CMD_WREG = 0x50, // Write to REG rrr 0101 rrrr (5xh)
9     CMD_SELF CAL = 0xF0, // Offset and Gain Self-Calibration 1111 0000 (F0h)
10    CMD_SELFOCAL = 0xF1, // Offset Self-Calibration 1111 0001 (F1h)
11    CMD_SELFGCAL = 0xF2, // Gain Self-Calibration 1111 0010 (F2h)
12    CMD_SYSOCAL = 0xF3, // System Offset Calibration 1111 0011 (F3h)
13    CMD_SYSGCAL = 0xF4, // System Gain Calibration 1111 0100 (F4h)
14    CMD_SYNC = 0xFC, // Synchronize the A/D Conversion 1111 1100 (FCh)
15    CMD_STANDBY = 0xFD, // Begin Standby Mode 1111 1101 (FDh)

```

```

16 CMD_RESET = 0xFE,           // Reset to Power-Up Values 1111    1110 (FEh)
17 };

```

Estos son los comandos vistos anteriormente.

Por ejemplo, en el caso del comando WAKE UP se debe ingresar la secuencia 0000 0000 que en hexadecimal es 0x00.

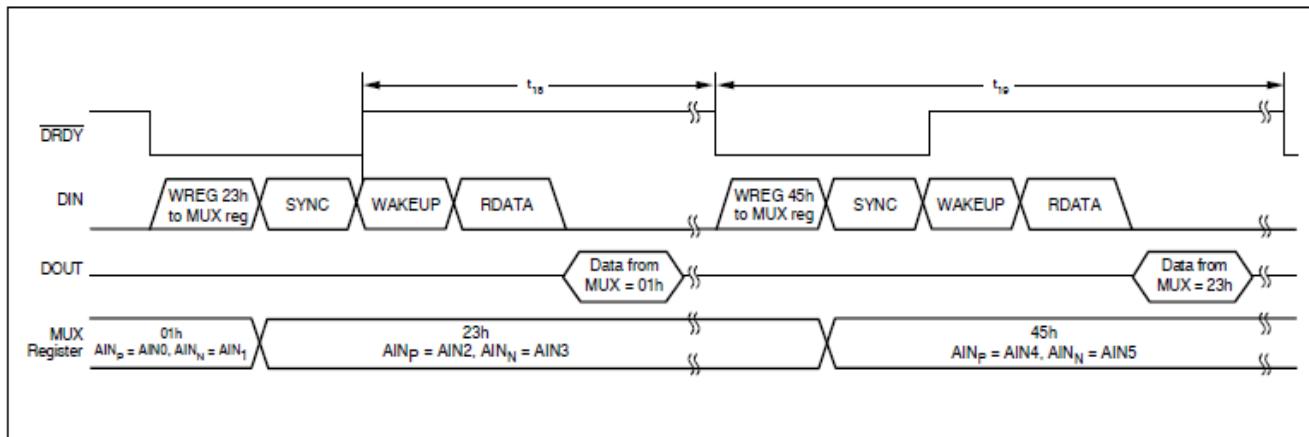
Si se conmutan las entradas, la frecuencia máxima de muestreo decae a 4374 (en realidad es aún menor), con lo que se obtienen ciclos (aproximados) de salida en microsegundos de:

```

1 static const uint32_t CYCLING_TROUGHPUT_USEC[ADS1256_DRATE_MAX] =
2 {
3     240,                      //aprox 1/(4374 Hz)
4     275,                      //aprox 1/3817
5     345,                      //aprox 1/3043
6     485,                      //aprox 1/2165
7     730,                      //aprox 1/1438
8     1254,                     //aprox 1/837
9     2303,                     //aprox 1/456
10    10714,                    //aprox 1/98
11    17797,                    //aprox 1/59
12    21000,                    //aprox 1/50
13    35000,                    //aprox 1/30
14    42000,                    //aprox 1/25
15    70000,                    //aprox 1/15
16    105000,                   //aprox 1/10
17    210000,                   //aprox 1/5
18    420000,                   //aprox 1/2.5
19 };
20
21 static const uint32_t SETTLING_TIME_T18_USEC[ADS1256_DRATE_MAX] =
22 {
23     210,
24     250,
25     310,
26     440,
27     680,
28     1180,
29     2180,
30     10180,
31     16840,
32     20180,
33     33510,
34     40180,
35     66840,
36     100180,
37     200180,
38     400180};

```

Como ejemplo del proceso completo de conversión y lectura:



*Figura 21

Observe que t_{19} es el periodo completo que abarca la escritura del registro MUX para comutar de entrada, el comando de sincronismo para sincronizar la lectura al clock, el comando para despertar una vez completada la sincronización y el comando para leer dicha entrada. Luego se obtienen los datos leídos por la entrada seleccionada en el ciclo anterior, por ello, se cambia el valor del registro MUX a 45h para utilizar las entradas AIN4(+) y AIN5(-) pero por DOUT se leerá lo obtenido en el momento de colocar el valor 23h en MUX y seleccionar las entradas AIN2(+) y AIN3(-). En este ejemplo se está usando el modo diferencial.

El tiempo t_{18} es solo el tiempo en que el pin DRDY se encuentra en alto, indicando que se está realizando una conversión y pasará a estado bajo cuando los datos de la entrada estén listos.

Por último, $\text{MASTER_CLOCK_PERIOD_USEC_TIMES_24} = 4$ se obtiene teniendo en cuenta el periodo del clock proporcionado por el cristal de 7.68 MHz.

```
1 static const uint64_t MASTER_CLOCK_PERIOD_USEC_TIMES_24 = 4; //24*0.13
```

Instrucciones provistas por AD-DA-WS-RPI.c

De la misma forma en que se describieron aquellos comandos más importantes, se hará un repaso de aquellas instrucciones más relevantes y que serán aquellas que se utilizarán para el desarrollo del programa de medición.

- **bsp_DelayUS(uint64_t micros)**

Realiza una espera activa de la cantidad de microsegundos especificados en su argumento. Esta función llama a `spi_delay_us(micros)` que se encuentra en `AD-DA-BCM.c`. Al mismo

tiempo esta función llama a `bcm2835_delayMicroseconds(micros)`; que es una función implementada en `bcm2835.c`.

Limitaciones de rendimiento en tiempo real

`bcm2835.h` es una biblioteca para programas de usuario. Tales programas no son parte del kernel y generalmente están sujetos a paginación e intercambio por parte de este mientras realiza otras acciones además de la ejecución del programa. Esto significa que no debe esperar obtener rendimiento en tiempo real o limitaciones de tiempo en tiempo real de dichos programas. En particular, no hay garantía de que [bcm2835_delay\(\)](#) y [bcm2835_delayMicroseconds\(\)](#) regresen exactamente después del tiempo solicitado. De hecho, dependiendo de otra actividad en el host, IO, etc., es posible que obtenga tiempos de demora significativamente más largos que los que solicitó. Por lo tanto, no espere obtener exactamente el tiempo de demora que solicita.

- **`bool DRDYIsLow(void)`**

Esta función, el único trabajo que realiza es retornar una variable booleana `DRDY_IS_LOW()`; definida en la biblioteca AD-DA-BCM.h como 0.

- **`bool DRDYIsHigh(void)`**

Retorna un valor distinto a `DRDY_IS_LOW()`, es decir, un 1.

- **`int ADS1256_WaitDRDY_LOW(void)`**

Esta función espera a que el pin DRDY se ponga en bajo. Llama a `returnWaitCondition(DRDYIsLow)` el cual espera a que ocurra una determinada condición, en este caso, que la función `DRDYIsLow` devuelva su valor booleano igual a 0.

Esta será la función que implementaremos para esperar a que el pin DRDY se ponga en bajo indicando los datos se encuentran disponibles.

- **`int ADS1256_WaitDRDY_HIGH(void)`**

Realiza la misma función que la anterior, pero espera a que el pin DRDY pase a estado alto indicando que comienza una conversión.

- **`int WaitCondition(bool (*f)(void))`**

Esta es la función que implementan las dos anteriores para realizar sus esperas. Recibe como argumento un puntero a una variable booleana (0 en caso de `ADS1256_WaitDRDY_LOW` y 1 en caso de `ADS1256_WaitDRDY_HIGH`). Estas dos funciones devolverán sus respectivos valores

cuando exista un cambio en el pin DRDY. WaitCondition ejecuta una espera activa a través de un bucle cuya duración es mayor que el periodo de t19 visto antes. Cada iteración del bucle llama a la función que realiza una espera de 1 microsegundo y pregunta si la variable que recibió como argumento cambio al valor esperado. De ser el caso, la función termina retornando un cero, indicando que finalizó correctamente. Si el bucle for se acaba, retorna -1.

- **Int ADS1256_ConfigureADC(ADS1256_GAIN_E _gain, ADS1256_DRATE_E _drate)**

Su función es configurar los parámetros del ADC. Se especifica la ganancia y la frecuencia de muestreo con las variables descritas anteriormente.

Su primera acción es determinar si, a través de la función ADS1256_WaitDRDY_LOW(); el pin DRDY se encuentra en estado alto y de ser el caso lo informa por el puerto serie retornando el valor -1.

Su próxima acción es configurar los registros del ADC:

El primero es el registro STATUS haciendo $buf[0] = (0 \ll 3) | (1 \ll 2) | (0 \ll 1)$, (notar que referencia al registro STATUS en la posición 0, ya que es el valor que se le asigna a dicho registro en la librería), que es equivalente a hacer ADS1256_WriteReg(REG_STATUS, $(0 \ll 3) | (1 \ll 2) | (0 \ll 1)$); lo que indica lo siguiente:

1. El bit 3 de orden se le asigna el valor cero, por lo que los datos saldrán en el orden de los bits más significativos primero.
2. El bit 2 en 1, por lo que la auto calibración está habilitada.
3. El bit 1 en 0, por lo que el buffer de entrada se mantendrá deshabilitado. Estos valores vienen por defecto.

Luego al registro MUX (1) es configurado con: $buf[1] = 0x08$. Esto significa que le es asignando el valor 0000 1000, colocando de esta forma en 1 el bit 3 NSEL3 y se utilizará el pin AINCOM como pin de referencia, ignorando los bits 2,1 y 0.

Luego es configurado el registro ADCON (2) con $buf[2] = (0 \ll 5) | (0 \ll 3) | ((uint8_t)_gain \ll 0)$. De esta forma se le indica que el clock de salida es 00 y está inhabilitado con lo que no se gastan recursos en un clock que no se usará. De lo contrario por defecto esta configurado para que este sea igual a la frecuencia de entrada del oscilador. Los bits 3 y 4 se configuran con 00 por lo que los sensores detectores de corriente para prueba de sensores están inhabilitados. Por últimos los bits 2, 1 y 0 se les dan los valores de ganancia pasados en el argumento de la función y con dicho valor se configura la ganancia del PGA. Por defecto viene en 1.

Por último, se configura el registro DRATE con $buf[3] = s_tabDataRate[_drate]$ pasándole como argumento la frecuencia de muestreo en muestras por segundo y utiliza la función s_tabDataRate para tomar el valor definido dentro de esta función en AD-DA-WS-RPI.h.

El próximo paso consiste en habilitar la comunicación serie. CS, pin de chip select, debe permanecer en bajo mientras se efectúe la comunicación, por lo que es llamada la función CS_ADC_0(); la cual indica al BCM2835 que coloque en alto dicho pin para iniciar la comunicación.

Luego es enviado por puerto serie el comando WREG para modificar efectivamente los registros del ADC, ya que los pasos anteriores solo guardaron en arreglos los valores que se escribirán en cada registro. Esto se hace con ADS1256_Send8Bit(CMD_WREG | 0) y con ADS1256_Send8Bit(0x03) con lo que se le indica que se escribirá en los registros empezando en el 0 y terminando en el 3.

Luego se mandan los valores almacenados en los arreglos:

```
ADS1256_Send8Bit(buf[0]); /* Set the status register */  
ADS1256_Send8Bit(buf[1]); /* Set the input channel parameters */  
ADS1256_Send8Bit(buf[2]); /* Set the ADCON control register,gain */  
ADS1256_Send8Bit(buf[3]);/* Set the output rate */
```

Por último, se cierra la comunicación serie con CS_ADC_1(), poniendo en 1 al pin chip select. Espera 50 uS y finaliza la función.

- **void ADS1256_WriteReg(uint8_t _RegID, uint8_t _RegValue)**

Esta función es utilizada para modificar los valores de registros y es esencial. Su argumento es el número que identifica al registro a escribir y el valor que será escrito.

Primero se debe habilitar la comunicación serie con CS_ADC_0(). Luego se envía el comando WREG con ADS1256_Send8Bit(CMD_WREG | _RegID), pasándole el número de registro a modificar y luego ADS1256_Send8Bit(0x00) como indicación de que solo modifique el registro especificado en RegID. Lo siguiente es el paso del valor a escribir con ADS1256_Send8Bit(_RegValue), y por último se cierra la comunicación con CS_ADC_1().

- **uint8_t ADS1256_ReadReg(uint8_t _RegID)**

Al igual que la función anterior, esta es utilizada para leer los bits de un registro. También se inicia la comunicación serie y luego se aplica ADS1256_Send8Bit(CMD_RREG | _RegID), indicando con el comando CMD_RREG que se leerá el registro RegID. Luego se debe esperar un determinado tiempo hasta que pueda obtenerse efectivamente los datos leídos, por lo que se coloca un delay con ADS1256_DelayDATA(). Al final se leen los datos con read = ADS1256_Receive8Bit(), cuya función ya fue explicada y se almacena el valor leído en read que será el valor de retorno de la función. Solo se necesitan 8 bits porque los registros de control poseen esta cantidad. Por último, se cierra la comunicación y se retorna el valor leído.

- **void ADS1256_WriteCmd(uint8_t _cmd)**

Sirve para enviar un byte de orden para ejecutar un comando. Inicia la comunicación serie, envía 8 bits por el puerto serie con la función ADS1256_Send8Bit(_cmd), pasándole como argumento el número que identifica al comando. Estos números fueron definidos en la librería AD-DA-WS-RPI.h.

- **void ADS1256_SetChannel(uint8_t channel)**

Esta función será de vital importancia para nuestro programa ya que nos permitirá cambiar de canal y leer las 8 entradas en modo individual. Para esto debe modificar el registro MUX.

Se debe pasar como argumento el canal que deseé seleccionar. La primera acción es llamar a la función ADS1256_RemapChannelIndex(channel). Esta función esta implementada para salvar un error en el código original, ya que cuando se referencia al canal 0 se obtienen los datos del canal 1 y el canal 7 es el canal 0. Por lo tanto, esta función realiza la reasignación correcta para que al llamar al canal 0 se referencie al 1, el canal 1 al 2 y así sucesivamente hasta el canal 7 que se le asigna el 0.

Si es pasado como argumento un número mayor a 7 no devolverá nada pues no hay más de 8 canales, empezando en el 0 y terminando en el 7.

Luego realiza ADS1256_WriteReg(REG_MUX, (_ch << 4) | (1 << 3)), con lo que coloca en 1 el bit 3, lo que permite que se ignoren los bits 2, 1 y 0 y se tome de referencia el pin AINCOM. Esto es así solo en modo individual, donde se usan los 8 canales como entradas de medición referidas al mismo pin común. Los bits del 4 al 7 son asignados con el valor que representa a cada canal y de esta forma se elige el canal deseado.

- **void ADS1256_SetDiffChannel(uint8_t channel)**

En este caso se implementa esta función para trabajar en modo diferencial. Por lo tanto, se podrán usar solo 4 canales ya que se une uno para la entrada de medición y otro canal para la referencia.

También usa ADS1256_RemapChannelIndex(channel) pero luego, si por ejemplo, el valor devuelto es 0, entonces se tomará como entrada positiva al canal 0 y como referencia al canal 1, si devuelve un 1 entonces se tomará como positiva al canal 2 y referencia al 3 y así sucesivamente.

Esto lo hace a través de ADS1256_WriteReg(REG_MUX, (6 << 4) | 7), lo que significa que escribirá en el registro MUX; del bit 4 al 7 se le coloca el valor 6 por lo que el canal 6 será el positivo y del bit 3 al 0 el número 7 con lo que el canal 7 será el negativo.

- **int32_t ADS1256_ReadData(void)**

Esta es una función base que luego será implementada por la función que finalmente se utilizará para leer los canales del ADC. El autor señala que no usemos esta función de forma directa para leer si no la función ADS1256_ReadAdcValues.

Primero se inicia la comunicación serie. Luego se envía el comando CMD_RDATA para indicarle al ADC que leerá de sus entradas. Luego debe esperar el delay necesario t6 para poder efectivamente leer los datos obtenidos, por lo que implementa DelayDATA.

Anteriormente se crea un arreglo de 8 bits de 3 casilleros. Lo que hará será leer los valores devueltos por el puerto serie que serán los datos leídos del canal seleccionado y los guardará en este arreglo. Para esto llama a buf[0] = ADS1256_Receive8Bit(), y luego repite el procedimiento con la posición 1 y 2. Se deben concatenar estas 3 posiciones para obtener efectivamente el valor de 24 bits. Antes de realizar la concatenación, castea las posiciones del arreglo a 32 bits sin signo.

La concatenación la realiza haciendo primero una and entre la primera posición del arreglo con 0x00FF0000 y previo a haber desplazado el valor del arreglo 16 lugares. Es decir:

0000.0000.0000.0000.0000.0000.xxxx.xxxx(original)

0000.0000.xxxx.xxxx.0000.0000.0000.0000(desplazado)

0000.0000.xxxx.xxxx.0000.0000.0000.0000 y

0000.0000.1111.1111.0000.0000.0000.0000 =

0000.0000.xxxx.xxxx.0000.0000.0000.0000

Luego realiza una operación OR binaria entre el resultado y las siguientes posiciones del arreglo. Para eso también debe desplazar el valor de la posición 1 en 8 lugares y la posición 0 no debe ser desplazada. Queda, por lo tanto:

0000.0000.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx

Luego cierra la comunicación.

Por último, determina si la AND entre el valor final y 0x00800000 da distinto de cero con lo que obtendría un número negativo, ya que el primer número sería un 1. Si la AND arroja un uno y por lo tanto el resultado es negativo, entonces concatena el resultado con 0xFF000000 para usar complemento a 2. Al final retorna el resultado.

- **int ADS1256_ReadAdcValues(uint8_t **Channels, int NbChannels, ADS1256_SCAN_MODE mode, int32_t **AdcVals)**

Esta es la función completa y definitiva que utilizaremos para leer los valores de entrada del ADC.

Se deben definir ciertas variables que usaremos en esta función. La primera es *Channels, el cual es un puntero a un arreglo que se utilizará para especificar qué canal queremos leer. Es

pasado a la función como argumento a través de un puntero doble, es decir, se hará referencia al arreglo a través de una referencia al puntero que contiene la dirección de este arreglo. NbChannels es el tamaño del arreglo de Channels que, en nuestro caso, será de 8 debido a los 8 canales que se leerán. mode hace referencia a si los canales se leerán en modo diferencial o individual. El último argumento de la función es un puntero a la variable donde guardaremos el valor leído. Esta también es pasada con un doble puntero a la función.

Lo primero que debe hacer esta función es crear un arreglo dinámico donde guardará los valores leídos de los canales. Para esto utiliza `*AdcVals = malloc(NbChannels * sizeof(int32_t))`. De esta forma el parámetro AdcVals variará de tamaño en función de la cantidad de canales que desee leer y las posiciones tendrán un tamaño de 32 bits.

A través de un bucle for irá avanzando por los canales hasta llegar al número 7.

Como ya se ha mencionado anteriormente, para poder leer los valores obtenidos, se debe esperar a que el pin DRDY se coloque en estado bajo. Por lo tanto, se llama a la función `ADS1256_WaitDRDY_LOW()` para esperar dicho suceso. Si el valor devuelto es distinto de 0 entonces lo indicará por el puerto serie y retornará el valor -1, indicando que ocurrió un error, ya que el tiempo de conversión excedió su límite. Luego se crea la variable CurChannel = `(*Channels + i)` que usará para referenciar el canal actual que se leerá.

*Notar que a la función se le debe pasar la dirección de memoria del puntero que contiene al arreglo Channels, ya que es un puntero doble. Al trabajar dentro de esta función con la variable *Channels se hace referencia a una posición dentro del arreglo que luego se incrementa en i. Luego con (*Channels + i) se hace referencia a la dirección de esa posición en particular.*

El siguiente paso es setear efectivamente el canal elegido. Para esto se pregunta si el modo especificado en el argumento es igual a `SINGLE_ENDED_INPUTS_8`. De ser el caso, entonces se llamará a `ADS1256_SetChannel(CurChannel)` para trabajar en modo individual y se le pasará como argumento el canal actual. En caso contrario, se trabajará en modo diferencial y se llamará a `ADS1256_SetDiffChannel(CurChannel)`.

Luego se espera a que se establezca definitivamente el canal elegido esperando 4 uS: `bsp_DelayUS(MASTER_CLOCK_PERIOD_USEC_TIMES_24)`.

Anteriormente se vio que el procedimiento para leer consistía en elegir el canal, ejecutar los comandos SYNC, WAKEUP y por último RDATA (Véase figura 21).

Por lo tanto, se ejecuta `ADS1256_WriteCmd(CMD_SYNC)`, que indicará que ejecute el comando SYNC; luego se esperan 4 uS y se ejecuta `ADS1256_WriteCmd(CMD_WAKEUP)`.

Para realizar el comando RDATA se llama a la función `int32_t RD = ADS1256_ReadData();`; de esta forma el valor leído será devuelto por esta función y almacenado en un entero de 32 bits RD.

Por último, se debe almacenar el valor leído en el arreglo dinámico AdcVals, por lo que se ejecuta `(*AdcVals + i) = RD`, es decir, primero aumentará en i la posición del arreglo AdcVals y luego se guardará el valor de RD en la dirección de memoria correspondiente a la posición seleccionada.

Ya que se trabaja con punteros, no es necesario retornar ningún valor, pero ya que la función está declarada para devolver un entero, en caso de que no haya ocurrido ningún error se retorna un 0.

- **int ADC_DAC_Init(int *id, ADS1256_GAIN_E aGain, ADS1256_DRATE_E aDrate)**

Función para iniciar la comunicación SPI del ADS1256 y el DAC 8552 y configurar la ganancia y frecuencia de muestreo de los datos de la conversión. Para ello utiliza las funciones de ADA-MOCKSPI.c.

Se le debe pasar como argumento: un puntero a un entero donde se guardará el ID del chip, la ganancia del PGA y la frecuencia de muestreo.

Primero llama a initBcm = spi_init(); esta función lo único que hace es escribir por el puerto serie “spi_init” y retornar un 1. Por lo tanto, se pregunta por este valor retornado y si es distinto de 1 lo indica por puerto serie y retorna -1 indicando error.

Luego llama a spiBegin = spi_begin() y spiPrepare = spi_init_adc_dac_board() que hacen exactamente lo mismo y retornan 1. También se obtiene el ID del chip a través de *id = ADS1256_ReadChipID() el cual debería ser igual a 3.

De lo contrario si ocurre algún error la función retorna:

- 1: spi_init falló.
- 2: spi_begin falló.
- 3: ADS1256_ReadChipID falló (id debería ser == 3).
- 4: spi_init_adc_dac_board falló.

Caso contrario, retorna un 0.

Por último, se llama a ADS1256_ConfigureADC(aGain, aDrate), función descripta anteriormente para setear la ganancia y la frecuencia de muestreo.

- **double ADS1256_AdcToMicroVolts(int32_t adcValue, double scalingFactor)**

Cuando se adquiere el valor leído de un canal, obtenemos un valor binario que debe escalarse por algún factor para su obtención en volts. Por eso, se debe pasar como argumento a esta función el valor leído de un canal almacenado en **adcValue** y el factor de escala. Por ejemplo, si desea el resultado en volts deberá pasar como factor de escala 1.0/1000000.0 ya que 1 V = 1000000uV.

Para esto hace doubleuV = scalingFactor * ((double)(adcValue)) * 5000000.0 / 8388608. Su primera acción es castear el valor de adcValue a double y luego lo multiplica por el factor de escala que deseamos y por el rango de voltaje que son los 5V máximos dividido por la cantidad de posibles combinaciones que se tienen con 23 bits, que es $2^{23}=8388608$ (23 bits efectivos libre de ruido).

Si lo desea en milivolts debería pasarle un factor de escala de 1000.0/1000000.0 y en microvoltios 1000000.0/1000000.0.

Por último, retorna este valor escalado.

- **int ADC_DAC_Close(void)**

Esta función cierra la comunicación SPI y pone al conversor en standby, que es el estado de bajo consumo cuando no se lo está utilizando.

Para esto llama a la función encargada de enviar los comandos por puerto serie ADS1256_WriteCmd(CMD_STANDBY). Produce una espera de 4 uS y llama a spi_end() que envía por puerto serie “spi_end” sin no retorna ningún valor. Luego closingBcm = spi_close() realiza la misma acción y devuelve un 1 si el cierre fue exitoso. De no ser así se indica que hubo un error al cerrar la comunicación y retorna el valor -1. Caso contrario, la función retorna un 0.

5.2.4 PROGRAMA DE MEDICIÓN

Se hará una descripción general del programa desarrollado para obtener las mediciones realizadas por la placa conversora y luego almacenarlas en archivos. El mismo fue desarrollado en lenguaje C y a continuación se detalla cada función con su respectivo diagrama de flujo.

- **Librerías utilizadas:**

```
/////////////////////////////Librerias////////////////////////////
#include "AD-DA-WS-RPI/AD-DA-WS-RPI.h"
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/types.h>
#include <signal.h>
#include <sys/wait.h>

//////////////////////Defines y Macros///////////////////
#define CS_ADC_0() printf("CS_ADC_0()\n") //Define para iniciar la comunicación serie
#define CS_ADC_1() printf("CS_ADC_1()\n") //Define para finalizar la comunicación serie
#define FIFO_PATH "/tmp/MI_FIFO"           //Nombre de la FIFO
```

- **Variables globales:**

```
//////////////////Declaración de variables globales/////////////////
```

```

int canal;                                //Canal de disparo seleccionado
int frequency;                            //Frecuencia de muestreo seleccionada
int N_post=0;                             //Cantidad de muestras post disparo seleccionada
int N_pre=0;                             //Cantidad de muestras pre-disparo seleccionada
float nivel;                               //Nivel de disparo seleccionado
int cantidad_archivos; //Cantidad máxima de archivos almacenables seleccionada
FILE *fp;                                  //Descriptor de archivo que usaremos para el archivo que
                                            //contiene los parámetros
FILE *file;                                //Descriptor de archivo que usaremos para el
                                            //archivo que contiene las lecturas
int RetCode = 0;                           //Variable de retorno para control
int flag_signal=0;                         //Variable bandera: toma el valor 1 cuando llega una señal
struct timeval start,current;             //Estructuras donde guardaremos el tiempo
uint8_t buf[4];                            //Arreglo para la habilitación del buffer de
                                            //entrada del ADC
char auxiliar[30]={" "};                  //Arreglo auxiliar utilizado en función parámetros
int DRATE_E;                             //Macro definida para la frecuencia de muestreo
int error=0;                               //Variable entera para control de errores

```

- **Función obtener:**

```
/////////////////////////////*Función "obtener"*////////////////////////
```

```

1 int obtener()
2 {
3     return atoi(auxiliar);           //Función que convierte una cadena de caracteres en
4                                     //un número entero
5 }

```

Función que permite implementar la función atoi dentro de una sentencia condicional. La función atoi, utilizada para convertir una cadena de caracteres pasada en su argumento en un número entero, falla al implementarla dentro de sentencias condicionales. Por esta razón, se decidió crear una función que es llamada en lugar de atoi y dentro de esta se implementa la función atoi, retornando el valor devuelto por ella.

- **Función frecuencia:**

Esta frecuencia permite, en base al parámetro de frecuencia obtenido del archivo de configuración, asignar a DRATE_E la macro correspondiente a la frecuencia seleccionada.

Diagrama de flujo:

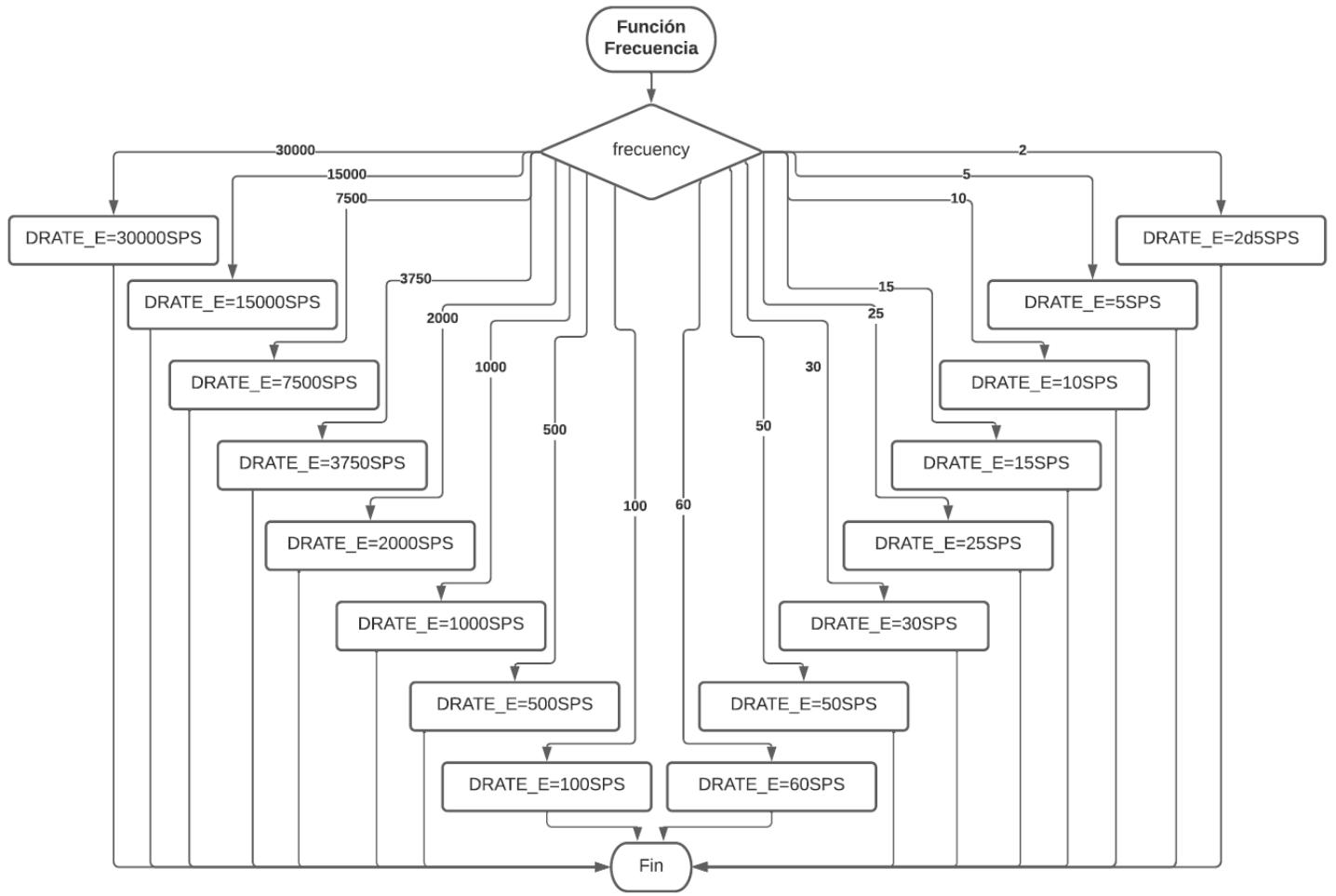


Figura 31: Diagrama de flujo de la función “frecuencia”.

Código fuente:

```

1 void frecuencia()
2 {
3     switch(frequency)
4     {
5         case 30000:
6             {DRATE_E=ADS1256_30000SPS; }
7             break;
8         case 15000:
9             {DRATE_E=ADS1256_15000SPS; }
10        break;
11        case 7500:
12            {DRATE_E=ADS1256_7500SPS; }
13            break;
14        case 3750:
15            {DRATE_E=ADS1256_3750SPS; }
16            break;
  
```

```

17     case 2000:
18         {DRATE_E=ADS1256_2000SPS; }
19         break;
20     case 1000:
21         {DRATE_E=ADS1256_1000SPS; }
22         break;
23     case 500:
24         {DRATE_E=ADS1256_500SPS; }
25         break;
26     case 100:
27         {DRATE_E=ADS1256_100SPS; }
28         break;
29     case 60:
30         {DRATE_E=ADS1256_60SPS; }
31         break;
32     case 50:
33         {DRATE_E=ADS1256_50SPS; }
34         break;
35     case 30:
36         {DRATE_E=ADS1256_30SPS; }
37         break;
38     case 25:
39         {DRATE_E=ADS1256_25SPS; }
40         break;
41     case 15:
42         {DRATE_E=ADS1256_15SPS; }
43         break;
44     case 10:
45         {DRATE_E=ADS1256_10SPS; }
46         break;
47     case 5:
48         {DRATE_E=ADS1256_5SPS; }
49         break;
50     case 2:
51         {DRATE_E=ADS1256_2d5SPS; }
52         break;
53     }
54 }
```

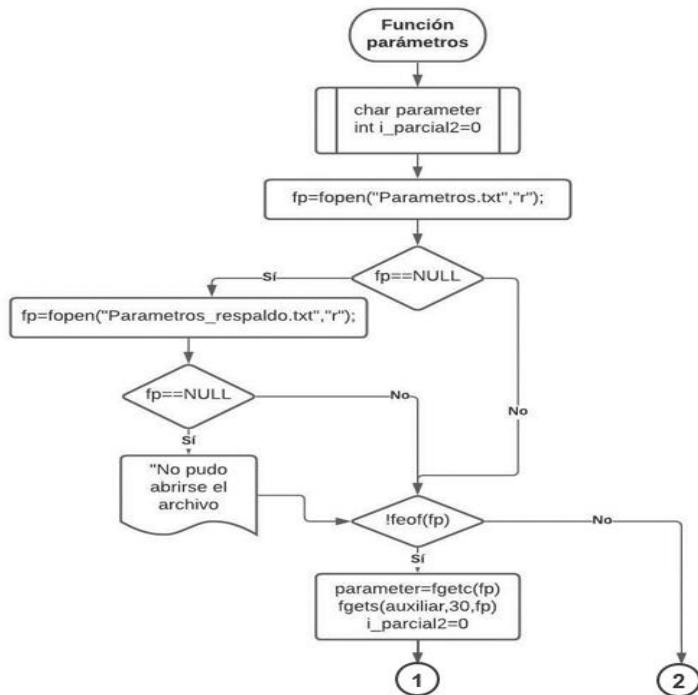
La macro ADS1256_30000SPS al igual que las demás están definidas en la biblioteca `\D-DA-WS-RPI.h` y representan a cada frecuencia.

La asignación para la frecuencia de muestreo de 2.5 SPS, se realiza en base a la condición de que el parámetro de frecuencia obtenido del archivo sea de 2 y no de 2.5 SPS. Esto es debido a que la función `atoi`, que realiza la conversión de la cadena de caracteres leída del archivo a un número entero, sólo devuelve valores enteros, es decir, solo retorna el número 2. Esto no es inconveniente ya que solo existe una frecuencia de muestreo seleccionable que posea un número 2 en la unidad.

- **Función parámetros:**

Previo a realizar cualquier medición, es necesario obtener los parámetros seleccionados por el usuario. Los mismos se encuentran en el archivo de configuración “Parametros.txt”. Esta función es la encargada de abrir dicho archivo, obtener los parámetros y asignarlos a las variables correspondientes.

Diagrama de flujo:



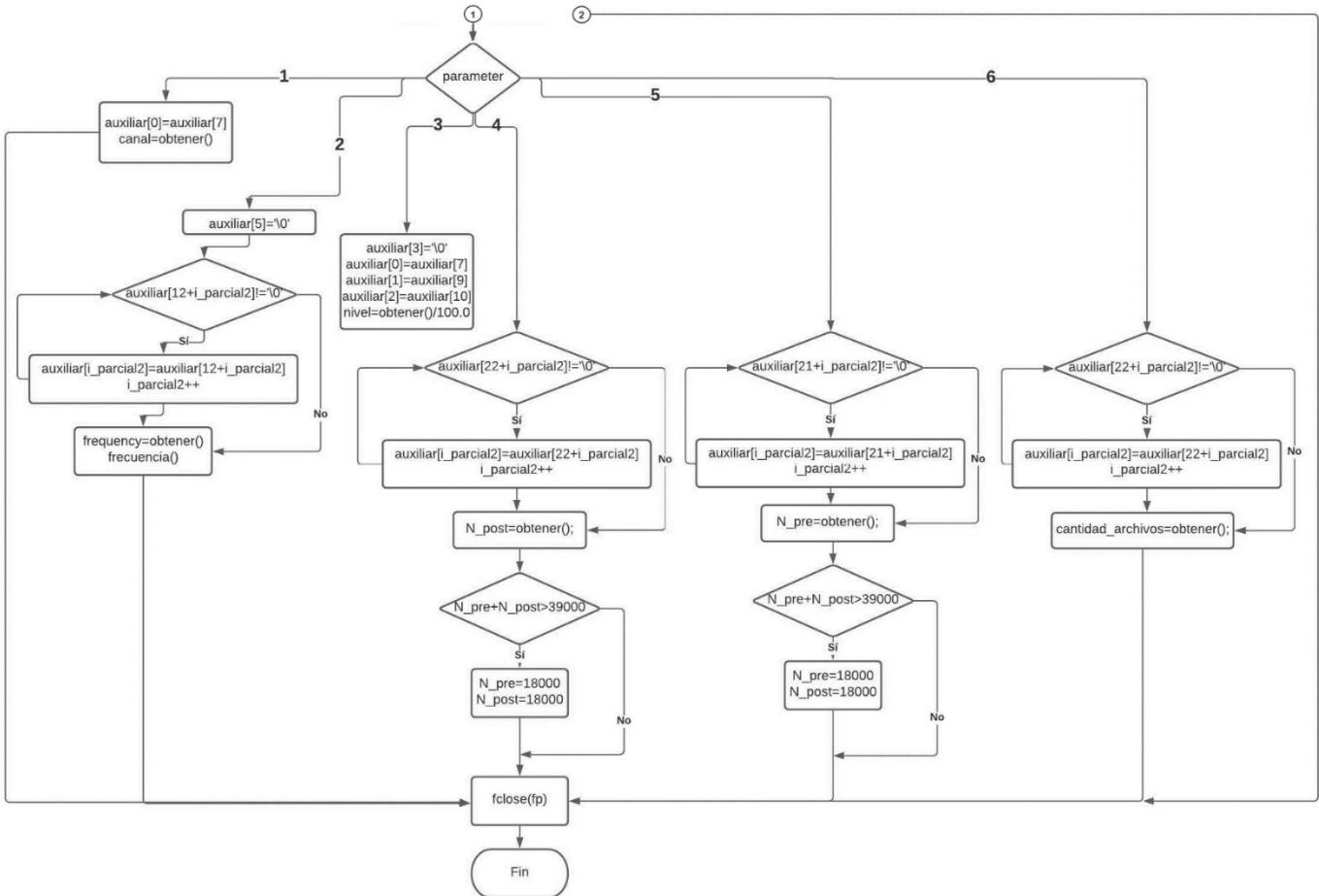


Figura 32: Diagrama de flujo de la función “parametros”.

Código fuente:

```

1 void parametros()
2 {
3     char parameter; //Carácter utilizado para la obtención del número de cada parámetro
4     int i_parcial2=0; //Entero auxiliar
5     fp=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/Parametros.txt","r"); //Abrimos el archivo de configuración
6     error=-1; //Si no se puede abrir el archivo Parametros.txt
7     if(fp==NULL) //Si fp=NULL no pudo abrirse el archivo Parametros.txt
8     { printf("No se pudo abrir archivo Parametros.txt\n");
9      fp=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/Parametros_respaldo.txt","r"); //Abrimos el archivo de configuración de respaldo
10    error=-2; //Si no se puede abrir el archivo Parametros_respaldo.txt
11    if(fp==NULL)
12    {printf("No se pudo abrir archivo Parametros_respaldo.txt\n");}
13 }

```

```

14     while(!feof(fp))           //Mientras no se llegue hasta el final del archivo se ejecuta
15     {
16         parameter=fgetc(fp);    //Obtenemos el primer carácter de cada fila
17         fgets(auxiliar,30,fp);  //Obtenemos la oración restante
18         i_parcial2=0;
19         switch(parameter)      //Según el valor del primer carácter de cada fila es el
19                           //parámetro a obtener
20     {
21         case '1':              //Parámetro canal
22         {
23             auxiliar[0]=auxiliar[7]; //El número de canal se encuentra en la posición
24                           //número 7 de la oración restante
25             canal=obtener();      //Obtenemos el valor numérico del canal
26             if(canal>7)
27             {canal=7;}
28             if(canal<0)
29             {canal=0;}
30         }
31         break;
32         case '2':              //Parámetro frecuencia
33         {
34             auxiliar[5]='\0';      //Hasta de la posición 5 borramos cualquier
34                           //contenido residual
35             while(auxiliar[12+i_parcial2]!='\0') //A partir de la posición 12 obtenemos
35                           //la frecuencia, hasta el final del
35                           //arreglo
36             {
37                 auxiliar[i_parcial2]=auxiliar[12+i_parcial2]; //Guardamos los caracteres
38                           //de la frecuencia en las
38                           //primeras posiciones del
38                           //arreglo auxiliar
39                 i_parcial2++;
40             }
41             frequency=obtener();   //Obtenemos el valor numérico de la frecuencia
41             frecuencia();          //Llamamos a la función frecuencia con la cual
41                           //asignamos a DRATE_E la macro correspondiente
42             }
42             break;
43             case '3':              //Parámetro nivel de disparo
44             {
45                 auxiliar[3]='\0';    //Hasta de la posición 3 borramos cualquier
45                           //contenido residual
46                 auxiliar[0]=auxiliar[7]; //Guardamos los caracteres de la frecuencia en las
46                           //primeras posiciones
47                 auxiliar[1]=auxiliar[9]; //del arreglo auxiliar, sin contar el punto decimal
48                 auxiliar[2]=auxiliar[10];
49                 nivel=obtener()/100.0; //Al dividir por 100 obtenemos el nivel de tensión
49                           //de disparo con dos posiciones decimales
50             }
51             break;

```

```

52     case '4': //Parámetro cantidad de muestras post trigger
53     {
54         while(auxiliar[22+i_parcial2]!='\0') //A partir de la posición 22 obtenemos
55             {                                la cantidad de muestras post disparo,
56                 auxiliar[i_parcial2]=auxiliar[22+i_parcial2]; //Guardamos los caracteres de
57                     i_parcial2++; //en las primeras posiciones del arreglo auxiliar
58             }
59         N_post=obtener(); //Obtenemos el valor numérico de la cantidad de
60             muestras post disparo //Limitamos la cantidad máxima de muestras a guardar a
61             39000 que es el tamaño máximo del buffer
62         {
63             N_pre=18000; //Tomamos un margen de precaución asignando 36000
64             N_post=18000; muestras en total
65         }
66     }
67     break;
68     case '5': //Parámetro cantidad de muestras pre trigger
69     {
70         while(auxiliar[21+i_parcial2]!='\0') //A partir de la posición 21 obtenemos
71             {                                la cantidad de muestras pre disparo,
72                 auxiliar[i_parcial2]=auxiliar[21+i_parcial2]; //Guardamos los caracteres de
73                     i_parcial2++; //en las primeras posiciones del arreglo auxiliar
74             }
75         N_pre=obtener(); //Obtenemos el valor numérico de la cantidad de
76             muestras pre disparo //Limitamos la cantidad máxima de muestras a
77             39000 que es el tamaño máximo del buffer
78         {
79             N_pre=18000; //Margen de precaución asignando 36000 muestras en total
80             N_post=18000;
81         }
82     }
83     break;
84     case '6': //Parámetro cantidad de archivos almacenables
85     {
86         while(auxiliar[22+i_parcial2]!='\0') //A partir de la posición 22 obtenemos
87             {                                la cantidad de archivos almacenables,
88                 auxiliar[i_parcial2]=auxiliar[22+i_parcial2]; //Guardamos los caracteres de
                     i_parcial2++; //en las primeras posiciones del arreglo auxiliar
             }

```

```

89     cantidad_archivos=obtener(); //Obtenemos el valor numérico de la cantidad de
90     archivos almacenables
91     }
92     break;
93   }
94   fclose(fp); //Cerramos el descriptor del archivo Parámetros.txt
95 }

```

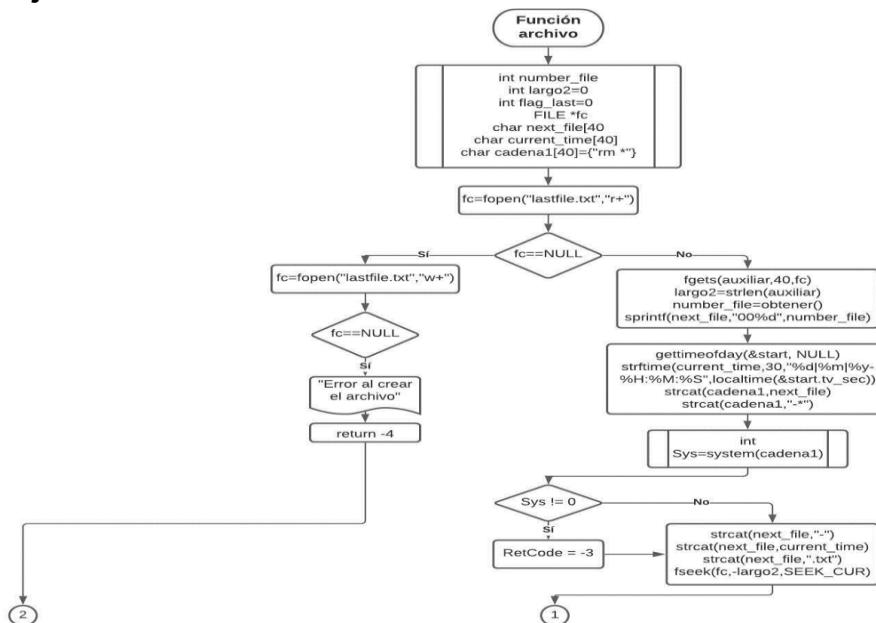
Consideraciones: en caso de no poder abrir el archivo “Parámetros.txt”, intentará abrir un archivo de configuración de respaldo “Parámetros_respaldo.txt”, el cual posee parámetros por defecto. Esto con el fin de que el programa pueda avanzar y tomar muestras, aunque los parámetros no hayan sido seleccionados por el usuario.

▪ Función archivo:

Una vez obtenidas las mediciones, las muestras deben almacenarse en un archivo numerado con la fecha y hora en la que ocurrió el disparo desencadenante de la generación del archivo. Esta función es la encargada de:

- Obtener el número del último archivo creado.
- Obtener la fecha y la hora.
- Si existe un archivo con la misma numeración, reemplazarlo por el nuevo archivo.
- Controlar que no se supere la cantidad máxima de archivos almacenables especificada por el usuario.
- Colocar el encabezado de los archivos con los parámetros de usuario, canales de medición y columna de tiempo.

Diagrama de flujo:



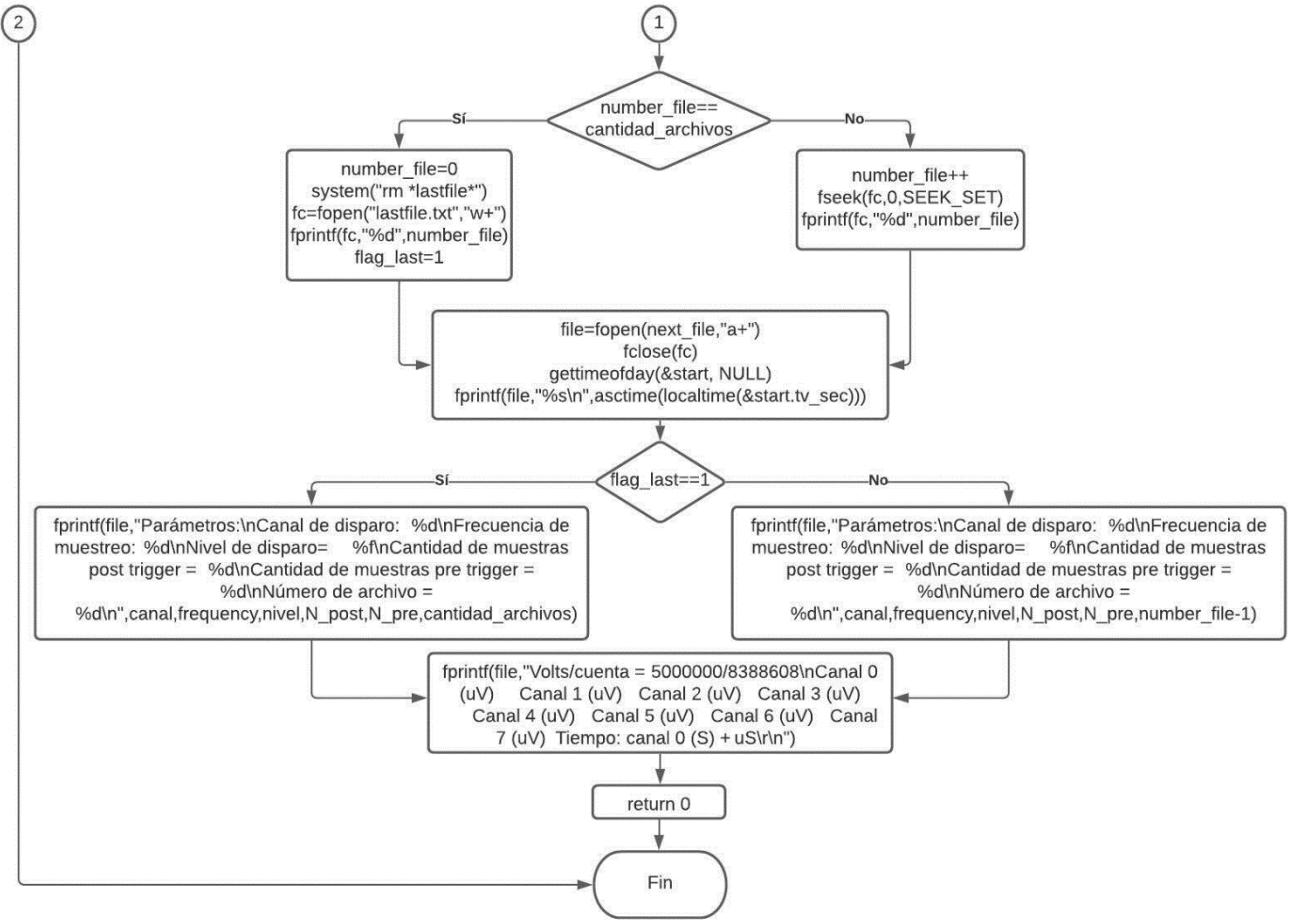


Figura 33: Diagrama de flujo de la función “archivo”.

Código fuente:

```

//////////*Función "archivo"*///////////

1 int archivo()           //Función para obtener el nombre del archivo a generar
2 {
3     int number_file;      //Entero con el número del archivo a generar
4     int largo2=0;          //Entero auxiliar con la longitud del arreglo obtenido
5     int flag_last=0;        //Variable bandera que toma el valor 1 cuando se alcanza la
                           //cantidad máxima de archivos almacenables
6     FILE *fc;             //Descriptor para el archivo con el número del archivo a generar
7     char next_file[40];     //Arreglo auxiliar donde guardaremos el nombre del
                           //siguiente archivo a generar
8     char current_time[40];   //Arreglo auxiliar donde guardaremos la fecha actual
                           //de generación del archivo
9     char cadena1[40]={ " rm -f ./home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
                           DAC/build/*" };           //Arreglo con sintaxis inicial del comando rm
10    char end_route[]={ "/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/" };

```

```

11 fc=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
DAC/build/lastfile.txt","r+");      //Abrimos el archivo en modo lectura/escritura
que contiene el número del último archivo creado o modificado
12 error=-3;                      //No pudo abrirse el archivo lastfile.txt en modo lectura
13 if(fc==NULL)                   //Si fc=NULL no pudo abrirse el archivo lastfile.txt
14 {
15 fc=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
DAC/build/lastfile.txt","w+");      //En caso de no existir el archivo, se crea y
luego se abre en modo escritura
16 if(fc==NULL)                   //Si fc=NULL no pudo crearse el archivo lastfile.txt
17 {printf("Error al crear el archivo\n");
18 error=-4;                      //No pudo abrirse el archivo lastfile.txt en modo escritura
19 return error;}      }
20 fgets(auxiliar,40,fc);          //Lectura del número del archivo a generar
21 largo2=strlen(auxiliar);        //Obtención de la longitud del arreglo auxiliar
22 number_file=obtener();          //Guardamos en number_file el número de archivo
23 sprintf(next_file,"00%d",number_file); //Guardamos el número de archivo en
                                         formato string en next_file
24 gettimeofday(&start, NULL);    //Obtención de la fecha y hora actual en la
                                         estructura start
25 strftime(current_time,30,"%d|%m|%y-%H:%M:%S",localtime(&start.tv_sec));
//Guardamos la fecha y hora actual en current_time en el formato especificado
entre comillas
26 strcat(cadena1,next_file);     //Concatenación de cadena1 con el número del
                                         archivo a generar
27 strcat(cadena1,"-*");
28 int Sys=system(cadena1);       //Con la función system escribimos por línea de
                                         comando el string contenido en cadena1
29 if (Sys != 0)                  //Si Sys es distinto de cero, no se encontró y por lo tanto
                                         no se eliminó ningún archivo con el nombre especificado
30 {   RetCode = -1;   }
31 strcat(next_file,"-");        //Conformamos el nombre del archivo concatenando el
                                         número, la fecha y hora y el formato del mismo
32 strcat(next_file,current_time);
33 strcat(next_file,".txt");
34 fseek(fc,-largo2,SEEK_CUR);    //Reubicamos el puntero del archivo al inicio
                                         del mismo
35 if(number_file==cantidad_archivos) //Si el número del último archivo coincide
                                         con el máximo especificado por el usuario,
                                         reiniciamos el conteo
36 {
37     number_file=0;
38     system("rm -f ./home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
                                         DAC/build/*lastfile*"); //Volvemos a crear el archivo
                                         lastfile.txt reiniciando el conteo del número de archivo
39     fc=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
                                         DAC/build/lastfile.txt","w+");
40     fprintf(fc,"%d",number_file);
41     flag_last=1;           //Levantamos la bandera flag_last
42 }
```

```

43     else          //Si no hemos llegado a la máxima cantidad de archivos simplemente
                  //sumamos 1 al número del último archivo
44     {
45         number_file++;
46         fseek(fc, 0, SEEK_SET);      //Reubicamos el puntero del archivo al inicio de
                                         //este
47         fprintf(fc, "%d", number_file); //Escribimos en lastfile.txt el número del
                                         //siguiente archivo a generar (no el actual)
48     }
49     strcat(end_route, next_file);   //Concatenación de la ruta de creación del
                                         //archivo con el nombre de este
50     file=fopen(end_route, "a+");    //Abrimos el archivo donde guardaremos las
                                         //lecturas. Si el archivo no existe se crea
51     fclose(fc);                   //Cerramos el archivo lastfile.txt
52     gettimeofday(&start, NULL); //Función para obtener la fecha y la hora actual
53     fprintf(file, "%s\n", asctime(localtime(&start.tv_sec)));
                                         //Imprimimos la fecha y la hora actual en el archivo generado
54     if(flag_last==1)           //Si flag_last=1 el encabezado del archivo debe poseer el
                                         //número de la cantidad máxima de archivos (esto se debe a
                                         //que anteriormente reiniciamos la cuenta)
55     {
56         fprintf(file, "Parámetros:\nCanal de disparo: %d\nFrecuencia de muestreo:
                                         %d\nNivel de disparo= %f\nCantidad de muestras post trigger =
                                         %d\nCantidad de muestras pre trigger =%d\nNúmero de archivo =
                                         %d\n", canal, frequency, nivel, N_post, N_pre, cantidad_archivos);
57     }
58     else           //Si flag_last=0 el encabezado del archivo debe poseer el número del
                                         //archivo actual que es igual a number_file-1
59     {
60         fprintf(file, "Parámetros:\nCanal de disparo: %d\nFrecuencia de muestreo:
                                         %d\nNivel de disparo= %f\nCantidad de muestras post trigger =
                                         %d\nCantidad de muestras pre trigger =%d\nNúmero de archivo =
                                         %d\n", canal, frequency, nivel, N_post, N_pre, number_file-1);
61     }
62     fprintf(file, "Microvolts/cuenta = 5000000/8388608\nCanal 0 (uV) Canal 1
(uV) Canal 2 (uV) Canal 3 (uV) Canal 4 (uV) Canal 5 (uV) Canal 6 (uV)
Canal 7 (uV) Tiempo: canal 0 (S) + us\r\n");
                                         //Encabezado de columnas para cada canal
63     return 0;
64 }

```

Consideraciones:

- El archivo *lastfile.txt* no contiene el número del último archivo creado, sino el siguiente. Esto se determinó en base a diversas pruebas y al orden en que se ejecutan las instrucciones.
- El archivo *lastfile.txt* primero se intenta su apertura en modo lectura/escritura en la línea 11. Sin embargo, si el archivo no existe o se encuentra dañado debe crearse nuevamente para

evitar que el programa se detenga o falle. Por esto se agregó la línea 15, en caso de error en la apertura del archivo, para crear el archivo.

- *La variable bandera flag_last permite reconocer la situación en la que el archivo generado posea la numeración de la cantidad máxima de archivos almacenables. De ser el caso, en lugar de utilizar number_file, se utiliza cantidad_archivos para generar el encabezado.*
- *La sobreescritura de archivos se realiza, primero preguntando si existe algún archivo con la misma numeración y, de ser este el caso, eliminando dicho archivo para ser reemplazado por el nuevo archivo. Esto se realiza así porque, aunque el número en el nombre de ambos archivos, el nuevo y el ya existente, sean los mismos, la fecha y hora son distintos, por lo que no se puede simplemente sobrescribir el contenido del archivo antiguo.*
- *La eliminación de archivos se realiza mediante el comando “rm”.*
- *Tener en cuenta que, si se modifica el parámetro de cantidad de archivos almacenables de una cantidad mayor a una menor, si existen los archivos de mayor numeración a la nueva cantidad especificada, estos no se eliminarán a menos que sean removidos manualmente accediendo a la placa. Para entender mejor la situación se dará un ejemplo. Suponga que originalmente la cantidad de archivos almacenables era 100 y se han generado los 100 archivos. Si se modifica esta cantidad a 50, al sobreescibirse los primeros archivos solo lo harán hasta el número 50, ya que esta es la nueva cantidad máxima a almacenar. Por lo tanto, los archivos cuyo número esté comprendido entre el 51 y el 100 no se sobreescibirán y quedarán almacenados hasta que el usuario acceda a la placa para eliminarlos o descargarlos.*

Función signal_handler:

Los parámetros modificables por el usuario se obtienen del archivo de configuración solo al inicio del programa. Luego se entra al bucle de medición donde se obtienen las muestras de forma constante en base a los parámetros obtenidos. Si el usuario modifica estos parámetros en el archivo de configuración, estos no serán leídos a menos que el programa sea reiniciado. Esto se implementó de esta manera ya que de lo contrario se debería abrir y leer el archivo de configuración cada vez que se obtenga una nueva muestra. Esto disminuiría enormemente la frecuencia de muestreo con la desventaja adicional de la pérdida significativa de muestras y el exceso de procesamiento.

Como una solución parcial se ha implementado el uso de una señal proveniente del programa por el cual la placa se comunica con el usuario a través de Telegram. De esta forma, si el usuario modifica remotamente los parámetros a través de la interfaz realizada en Telegram, posee la opción de aplicar los cambios en el momento que lo deseé. Se envía una señal al programa de medición el cual, al recibirla, entra en el manejador de la señal, poniendo en 1 una variable bandera. De esta forma, en el bucle de medición se entra en una estructura condicional donde se vuelven a obtener los parámetros y se aplican a las configuraciones correspondientes.

Código fuente:

```
/////////*Manejador de señal*//////////
1 void signal_handler(int unusable)          //Manejador de la señal proveniente del
                                             //programa de Telegram
2 {
3     unusable=0;                          //Variable sin uso necesaria para el correcto
                                             //funcionamiento del manejador
4     printf("\nunusable=%d\n",unusable);
5     flag_signal=1; //Levantamos la bandera flag_signal indicando que llegó una
6 señal }
```

Declaración en el main:

```
signal(SIGUSR1, signal_handler);

//Si llega una señal SIGUSR1, se atiende en el manejador de señal
signal_handler
```

Aplicación de los parámetros:

```
if(flag_signal==1) //Si flag_signal=1, llegó una señal y deben aplicarse los
                    //nuevos parámetros
{
    parametros();                      //Se obtienen los nuevos parámetros
    level_triggered=nivel*8388608/5; //Conversión de volts a número combinacional
    N_total=N_pre+N_post;           //Obtención del total de muestras a almacenar
    Init = ADC_DAC_Init(&Id, ADS1256_GAIN_1, DRATE_E); //Inicializamos el ADC. En
    Id obtenemos el numero de identificación del chip que debería ser igual a 3.
    if (Init != 0)                  //Elegimos ganancia igual a 1 y la frecuencia de muestreo
        en muestras/segundo
    { RetCode = -5; //Si no pudo reiniciarse el ADC, el valor de retorno valdrá -5
    }
    i_general=0;                   //Reinicio del conteo general del buffer
    i_parcial=-1;                 //Reinicio del conteo parcial
    flag_signal=0;                 //Bajada de la bandera flag_signal
}
```

Consideraciones:

- La variable ‘unusable’ dentro del manejador de la señal no se utiliza, sin embargo, es necesaria debido a que el manejador de la señal debe poseer argumento y este debe ser utilizado dentro del manejador para su correcta compilación y ejecución.
- Se decidió aplicar los cambios en los parámetros dentro del bucle de medición y no dentro del manejador por dos razones principales. La primera es que algunas de las variables requeridas para la configuración del ADC son declaradas dentro del main y no como variables globales. La segunda y más importante es que cuando arriba una señal proveniente de otro proceso, se interrumpe el programa principal y se ejecuta el manejador de la señal.

Esta situación se ilustra a continuación:

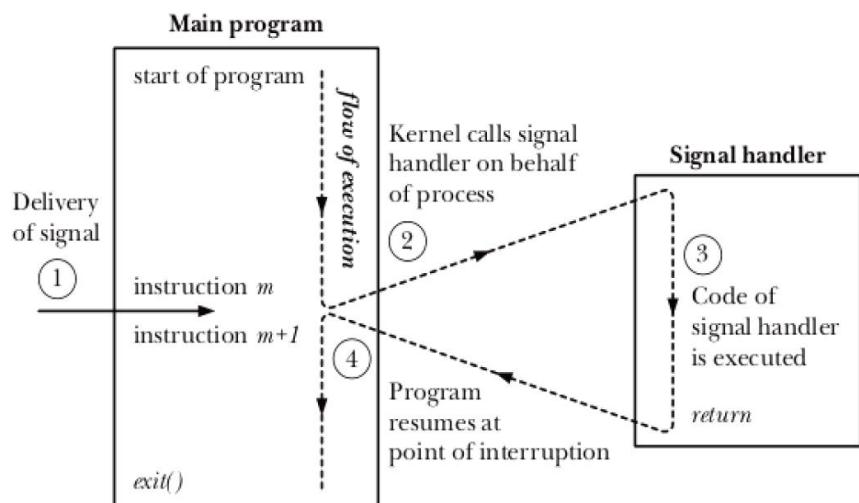


Figura 34: Proceso de atención de una señal.

Esto trae como consecuencia que, si al momento de aplicar los cambios e iniciar nuevamente la placa conversora para que los cambios sean aplicados, esta se encuentra midiendo, se produce un error que cuelga al programa. Debido a que la mayor parte del tiempo el programa principal se encuentra midiendo de forma constante, este error posee grandes probabilidades de ocurrencia.

En cambio, al aplicar los cambios dentro del bucle de medición antes de medir, nunca se aplican los cambios al mismo tiempo en que la placa conversora se encuentra tomando muestras.

▪ Función lectura:

Esta función es la encargada de ejecutar las instrucciones necesarias para la toma de mediciones. Es, por lo tanto, la función más importante. Está compuesta de las mismas instrucciones que la función ADS1256_ReadAdcValues, sin embargo, esta implementada con matrices en lugar de arreglos dinámicos. Esto se realizó de esta manera por 3 razones:

1. Son más seguros ya que al no haber redimensionamientos, el riesgo de errores es menor. También existen menos problemas en cuanto a que datos válidos sean eliminados por error.
2. Son más claros: no hay equívocos en cuanto al número de elementos que los componen y son más fáciles de seguir.
3. Ocupan menos recursos del procesador, el cual tiene que gestionar su contenido, pero no cambia el de dimensiones. Esta es la razón principal, ya que al ocupar menos recursos del ordenador se gana en velocidad, evitando así una reducción de las velocidades de muestreo.

Como contrapartida se pierde eficiencia en la gestión de memoria al poseer posiciones de memoria que no se utilizan.

Diagrama de flujo:

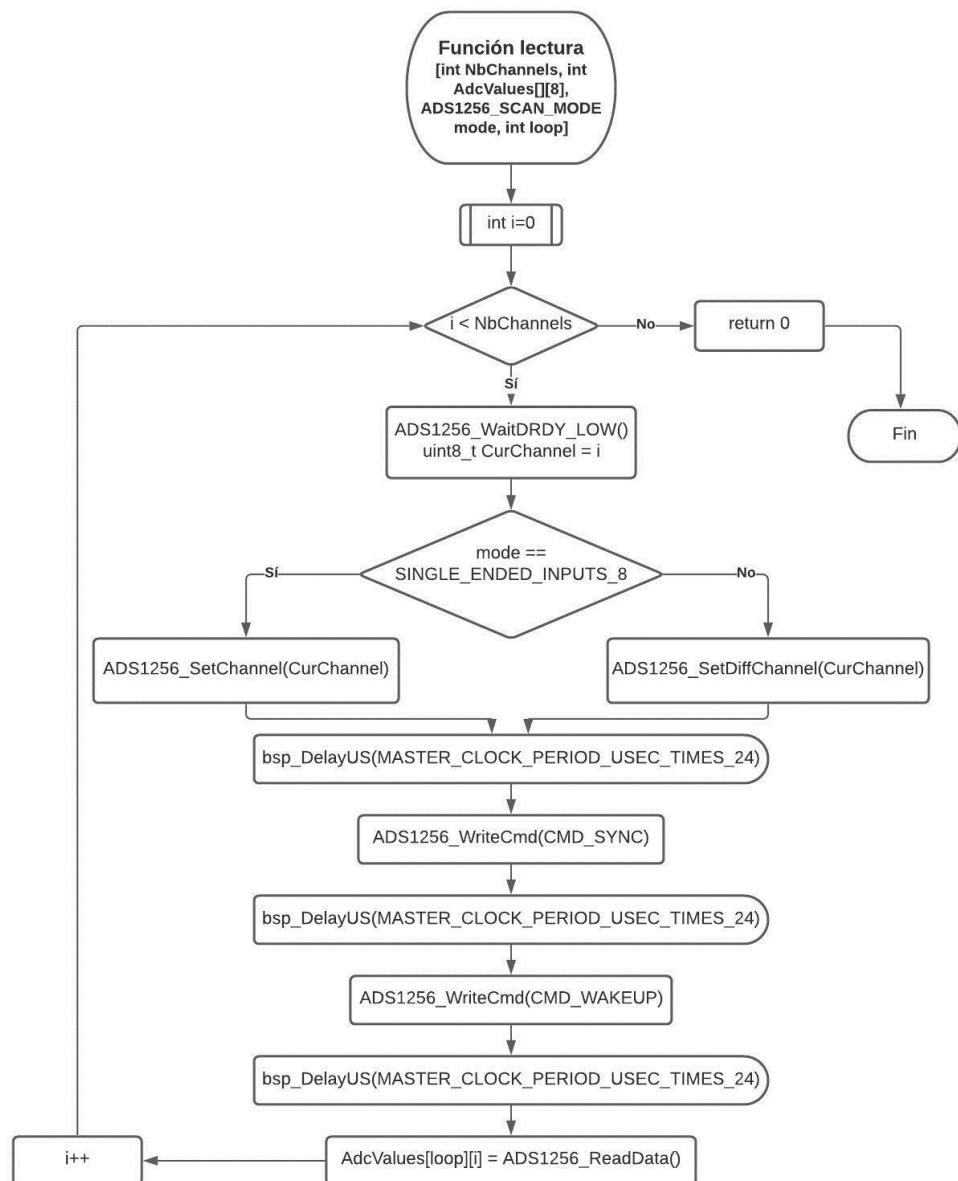


Figura 35: Diagrama de flujo de la función “lectura”.

Código fuente:

```

/////////*Función "lectura"*/////////
1 int lectura(int NbChannels, int AdcValues[][], ADS1256_SCAN_MODE mode, int loo

```

```

2 {
3     int i;
4     for (i = 0; i < NbChannels; i++)    //Con el bucle for recorremos los 8 canales
5     {
6         ADS1256_WaitDRDY_LOW();        //Espera a que el pin DRDY se ponga en bajo lo que
7         //indica que la conversión anterior terminó y puede iniciar la nueva conversión
8         uint8_t CurChannel = i;
9         if (mode == SINGLE_ENDED_INPUTS_8)          //Modo de 8 entradas individuales
10            ADS1256_SetChannel(CurChannel);           //Seteo del canal
11        else
12            ADS1256_SetDiffChannel(CurChannel);
13        bsp_DelayUS(MASTER_CLOCK_PERIOD_USEC_TIMES_24); //Tiempos de espera
14        //necesarios para que se establezcan correctamente los parámetros
15        ADS1256_WriteCmd(CMD_SYNC);                //Comando de sincronismo
16        bsp_DelayUS(MASTER_CLOCK_PERIOD_USEC_TIMES_24);
17        ADS1256_WriteCmd(CMD_WAKEUP);             //Junto con CMD_SYNC son comandos
18        //necesarios antes realizar la medición
19        bsp_DelayUS(MASTER_CLOCK_PERIOD_USEC_TIMES_24);
20        AdcValues[loop][i] = ADS1256_ReadData();    //Lee la entrada seleccionada y
21        //almacena la muestra en una posición del arreglo AdcValues
22    }
23    return 0;
24 }

```

- **Función main:**

Declaración de variables:

```

int main(void)
{
    signal(SIGUSR1, signal_handler);      //Si llega una señal SIGUSR1, se atiende en
                                         //el manejador de señal signal_handler

/////////////////////////////DEFINICIÓN DE VARIABLES////////////////////////////
    int NChannels = 8;                  //Cantidad de canales a leer
    int MainLoop = 0;                   //Variable a usar en caso de que no deseemos que el
                                         //programa se ejecute de forma indefinida sino n veces

    int j_general;                    //Entero auxiliar
    int flag_triggered=0;             //Variable bandera que toma el valor 1 cuando la
                                         //medición supera el valor umbral

    int i_referencia;                //Entero auxiliar que usaremos de referencia al
                                         //guardar las muestras en el archivo
    int pid_m;                       //Entero donde obtendremos el PID del proceso
    int err;                          //Entero de retorno al crear la FIFO
    int fifo_d;                      //Descriptor de la FIFO abierta
    int level_triggered=0;            //Variable donde guardaremos la combinación
                                         //equivalente al nivel seleccionado
    int N_total=0;                   //N será igual a la cantidad total de muestras a
                                         //guardar en el archivo, por lo tanto, N=post+pre
    int Init;                         //Variable de retorno al iniciar el ADC

```

```

int Id = 0; //Variable donde se guarda el identificador de la placa
int i_general=0; //Entero auxiliar
int i_parcial=-1; //Entero auxiliar (se declara en -1 para tener en cuenta la muestra que produjo el disparo)
char mypid[20]; //Arreglo auxiliar con el PID del proceso a enviar por la FIFO

```

Inicio de prueba:

Obtención de parámetros, inicio de la comunicación SPI y creación y escritura de la estructura de comunicación FIFO.

Diagrama de flujo:

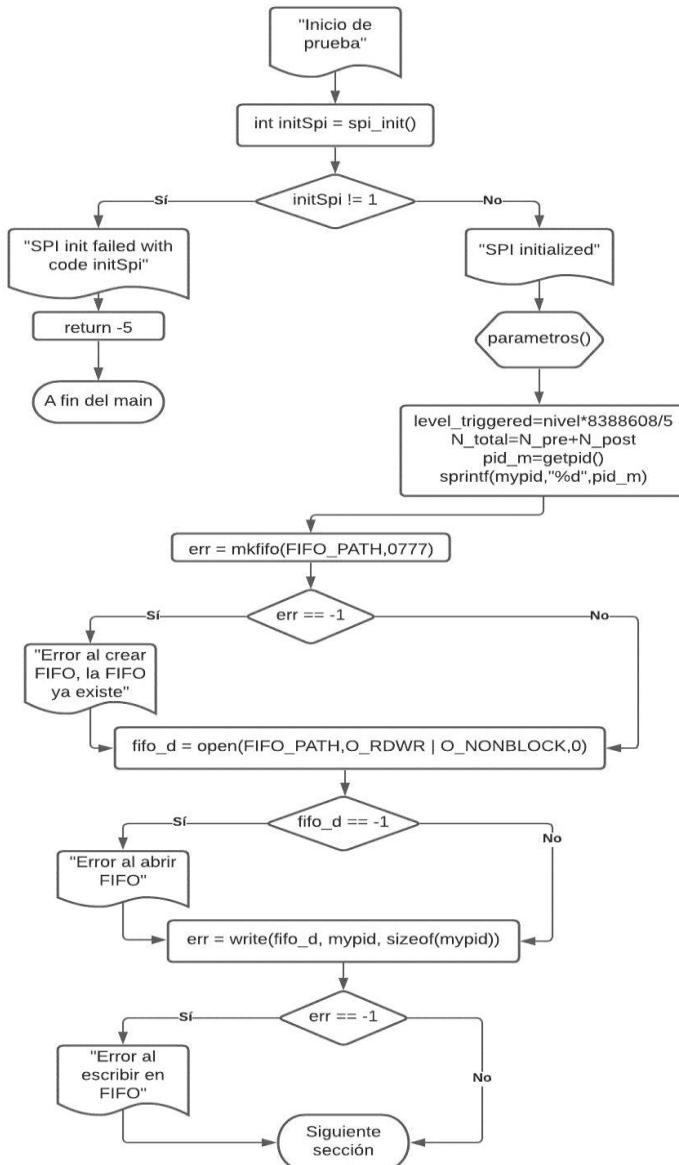


Figura 36: Diagrama de flujo del inicio de prueba.

Código fuente:

```
/////////////////////////////INICIO DE PRUEBA//////////////////////////  
  
1 printf("Iniciando prueba\r\n");  
2 int initSpi = spi_init(); //Inicializamos la comunicación SPI  
3 if (initSpi != 1) //Si initSpi es distinto de 1, no se pudo iniciar la  
    //comunicación serie con la placa  
4 {  
5     printf("SPI init failed with code %d\r\n", initSpi);  
6     error=-5; //No pudo iniciarse la comunicación serie  
7     return error;  
8 }  
9 printf("SPI initialized\r\n");  
10 parametros(); //Obtención de los parámetros  
11 level_triggered=nivel*8388608/5; //Conversión de volts a número  
    //combinacional del nivel de disparo  
12 N_total=N_pre+N_post; //La cantidad total de muestras a almacenar  
13 pid_m=getpid(); //Obtención del PID del proceso  
14 sprintf(mypid,"%d",pid_m); //Conversión a arreglo de caracteres del PID del  
    //proceso  
15 err = mkfifo(FIFO_PATH,0777); //Creamos la FIFO con los permisos  
    //especificados en el segundo argumento  
16 if(err == -1) //Si la FIFO ya existe porque no pudo eliminarse, err  
    //toma el valor -1  
17 {printf("\nError al crear FIFO, la FIFO ya existe\n");  
18 RetCode=-2;  
19 fifo_d = open(FIFO_PATH,O_RDWR | O_NONBLOCK,0);  
    //Apertura de la FIFO en modo lectura/escritura. Debe ser no bloqueante  
20 if(fifo_d == -1) //Si open devuelve -1 se produjo un error al abrir la FIFO  
21 {printf("\nError al abrir FIFO\n"); //caso contrario devuelve el descriptor de  
    //archivo de la FIFO  
22 RetCode=-3;  
23 err = write(fifo_d, mypid, sizeof(mypid));  
    //Escribimos en la FIFO el PID del proceso  
24 if(err == -1) //Si err=-1 se produjo un error al escribir en la FIFO  
25 {printf("\nError al escribir en FIFO\n");  
26 RetCode=-4;
```

Consideraciones:

- En la línea 7, se produce un retorno de la función main con el valor -5, finalizando así el programa, debido a que de no lograr establecer la comunicación SPI con la placa conversora no se podrán obtener las muestras y no se podrá cumplir el objetivo fundamental del programa.
- La obtención del PID (process ID) en la línea 12 se requiere para ser enviado al programa de acceso remoto por Telegram, ya que este último requiere el PID del proceso al que desea enviar la señal para la aplicación de los cambios en los parámetros de configuración.

- Por lo mismo explicado anteriormente, como mecanismo de comunicación entre procesos, de forma tal que el programa de acceso remoto conozca el PID del proceso de medición, se optó por el mecanismo FIFO (First Input, First Output). Por ellos, primero debe crearse la FIFO, abrirse y luego escribir en ella el PID del proceso.

Configuración del ADC y habilitación del buffer de entrada:

Inicialización del ADC, pasándole como argumento la ganancia y la frecuencia de muestreo y obteniendo el ID del circuito integrado.

Código fuente:

```
/////////////////////////////CONFIGURACIÓN DEL ADC///////////////////////////
1 Init = ADC_DAC_Init(&Id, ADS1256_GAIN_1, DRATE_E); //Inicializamos el ADC. En
Id obtenemos el numero de identificación del chip que debería ser igual a 3.
2 if (Init != 0) //Elegimos ganancia igual a 1 y la frecuencia de
muestreo en muestras/segundo
3 {
4     error = -6; //Si Init es distinto de 0, no pudo iniciarse el
ADC y retornamos el valor -6.
5     return error;6 }
6 printf("ADC_DAC_Init\r\n");
//////////////////Habilitación del buffer de entrada/////////////////
/*Comentar las líneas resaltadas si desea deshabilitar el buffer de entrada*/
7 buf[0] = (0 << 3) | (1 << 2) | (1 << 1);
8 buf[1] = 0x08;
9 buf[2] = (0 << 5) | (0 << 3) | ((uint8_t) ADS1256_GAIN_1 << 0);
10 CS_ADC_0(); /* SPI cs = 0 */
11 ADS1256_Send8Bit(CMD_WREG | 0); /* Write command register, send the register
address */
12 ADS1256_Send8Bit(0x03); /* Register number 4,Initialize the number -1*/
13 ADS1256_Send8Bit(buf[0]); /* Set the status register */
14 ADS1256_Send8Bit(buf[1]); /* Set the input channel parameters */
15 ADS1256_Send8Bit(buf[2]); /* Set the ADCON control register,gain */
16 ADS1256_Send8Bit(buf[3]); /* Set the output rate */
17 CS_ADC_1(); /* SPI cs = 1 */
18 printf("init done !\r\n");
```

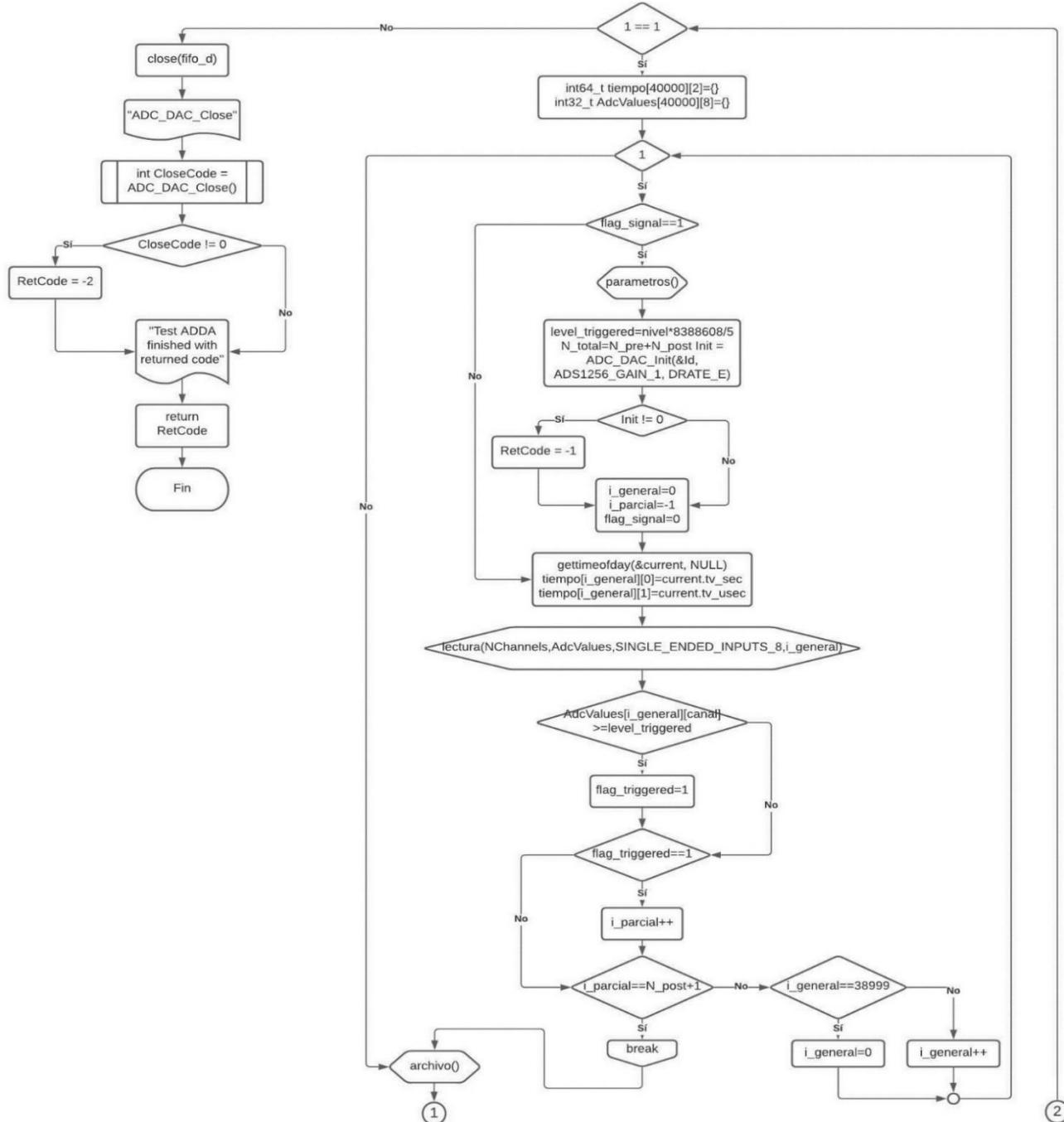
Consideraciones: Para deshabilitar el buffer analógico de entrada del ADS1256 debe comentar de la línea 7 a la 17, las líneas resaltadas.

Bucles principales:

Debido a que el programa debe muestrear de forma constante y guardar las muestras en archivos, se han implementado 2 bucles principales. El primero y de mayor jerarquía es el encargado de reiniciar los arreglos donde se guardan las muestras y las estampas de tiempo,

llamar a la función generadora de archivos, y escribir las muestras en dicho archivo. El segundo bucle corre dentro del primero y es el bucle encargado de medir en forma constante y aplicar los cambios a los parámetros de usuario cuando arriba una señal.

Diagrama de flujo:



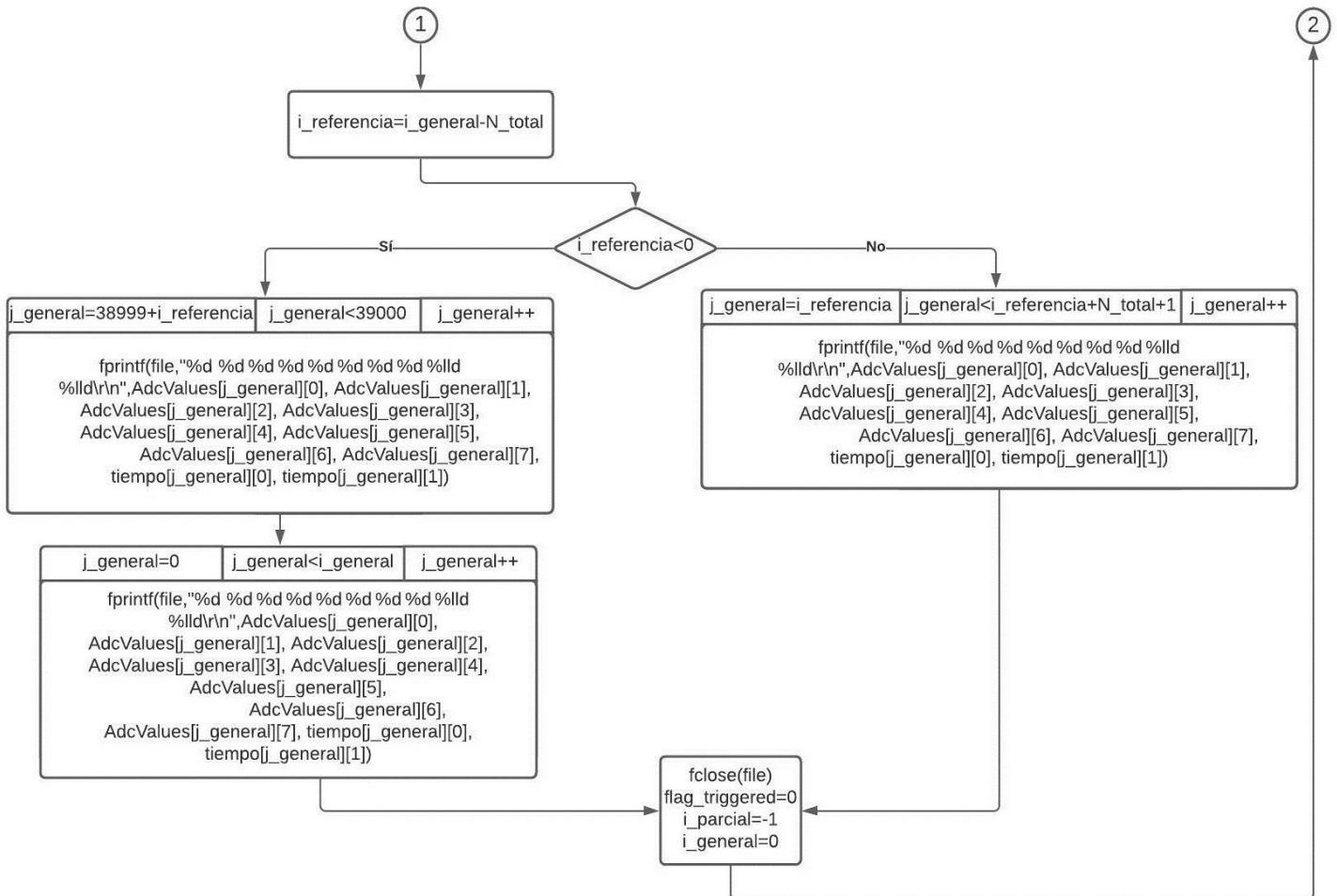


Figura 37: Diagrama de flujo de los bucles principales.

Código fuente:

```

//////////////////////////////BUCLE PRINCIPAL///////////////////////////
1 while (1 == 1)
2 {
3     int64_t tiempo[40000][2]={};           //Matriz donde guardaremos el tiempo en segundos y
                                         //microsegundos
4     int32_t AdcValues[40000][8]={};         //Arreglo donde guardaremos los valores
                                         //leidos de cada canal(se rellena con los argumentos pasados en ReadAdcValues)

//////////////////////////////BUCLE SECUNDARIO///////////////////////////
5 while(1)
6 {
7     if(flag_signal==1)                  //Si flag_signal=1, llegó una señal y deben aplicarse
                                         //los nuevos parámetros
8     {
9         parametros();                  //Se obtienen los nuevos parámetros
10        level_triggered=nivel*8388608/5; //Conversión de volts a número combinacional
11        N_total=N_pre+N_post;          //Obtención del total de muestras a almacenar

```

```

12  Init = ADC_DAC_Init(&Id, ADS1256_GAIN_1, DRATE_E);      //Inicializamos el ADC. En
   Id obtenemos el número de identificación del chip que debería ser igual a 3.
13  if (Init != 0)           //Elegimos ganancia igual a 1 y la frecuencia de muestreo
   en muestras/segundo
14  {
15    RetCode = -5;        //Si no pudo reiniciarse el ADC, el valor de retorno valdrá -5
16  }
17  i_general=0;          //Reinicio del conteo general del buffer
18  i_parcial=-1;         //Reinicio del conteo parcial
19  flag_signal=0;         //Bajada de la bandera flag_signal
20  }
21  gettimeofday(&current, NULL);           //Obtención del tiempo actual en la
   estructura current
22  tiempo[i_general][0]=current.tv_sec;    //Guardamos los segundos en la primera
   columna y la fila especificada por i_general
23  tiempo[i_general][1]=current.tv_usec;    //Guardamos los microsegundos en la segunda
   columna y la fila especificada por i_general
24  lectura(NChannels,AdcValues,SINGLE_ENDED_INPUTS_8,i_general);
   //Pasamos el arreglo con los canales a leer, la cant. de canales,
   el modo (singular o diferencial) y el arreglo donde guardar los valores leidos
25  if(AdcValues[i_general][canal]>=level_triggered) //Si la medición es mayor o
   igual al nivel especificado, levantamos la bandera flag_triggered
26  {
27    flag_triggered=1;
28  }
29  if(flag_triggered==1)      //Si la bandera flag_triggered=1, incrementamos el
   contador parcial
30  {
31    i_parcial++;
32  }
33  if(i_parcial==N_post+1)    //Si el contador parcial alcanza la cantidad de
   muestras post disparo + 1, salimos del bucle de medición
34  {
35    break;
36  }
37  if(i_general==38999)       //Si se alcanza la última posición del buffer,
   reiniciamos el contador general
38  {
39    i_general=0;
40  }
41  else                      //Caso contrario, incrementamos en 1 la posición del buffer
42  {
43    i_general++;
44  }
45  }
46  archivo();    //Obtenemos y abrimos el archivo donde guardaremos las mediciones
47  i_referencia=i_general-N_total;        //Generación de la variable que tomaremos de
   referencia
48  if(i_referencia<0)      //Si i_referencia es negativo, estamos en el caso en el que
   las muestras se encuentran tanto al final como al principio del buffer
49  {

```

```

50 for(j_general=38999+i_referencia;j_general<39000;j_general++) //Recorremos el
   buffer desde la posición final de este menos i_referencia hasta la posición final
51 {
52   fprintf(file,"%d %d %d %d %d %d %d %d %d %ld
      %ld\r\n",AdcValues[j_general][0], AdcValues[j_general][1],
      AdcValues[j_general][2], AdcValues[j_general][3], AdcValues[j_general][4],
      AdcValues[j_general][5], AdcValues[j_general][6], AdcValues[j_general][7],
      tiempo[j_general][0], tiempo[j_general][1]);
   //Guardamos en el archivo la muestra obtenida de cada canal con su estampa de
   tiempo en segundos y microsegundos
53 }
54 for(j_general=0;j_general<i_general;j_general++) //Recorremos el buffer
   desde la posición inicial hasta la posición con la última muestra tomada
55 {
56   fprintf(file,"%d %d %d %d %d %d %d %d %d %ld
      %ld\r\n",AdcValues[j_general][0], AdcValues[j_general][1],
      AdcValues[j_general][2], AdcValues[j_general][3], AdcValues[j_general][4],
      AdcValues[j_general][5], AdcValues[j_general][6], AdcValues[j_general][7],
      tiempo[j_general][0], tiempo[j_general][1]);
   //Guardamos en el archivo la muestra obtenida de cada canal con su estampa de
   tiempo en segundos y microsegundos
57 }
58 }
59 else //Si i_referencia es positivo, estamos en el caso en el que las muestras se
   encuentran en la zona intermedia del buffer
60 {
61   for(j_general=i_referencia;j_general<i_referencia+N_total+1;j_general++)
      //Recorremos el buffer desde i_referencia (1ra muestra pre disparo)
62   {
      hasta la última muestra tomada (última muestra post disparo)
63   fprintf(file,"%d %d %d %d %d %d %d %d %d %ld
      %ld\r\n",AdcValues[j_general][0], AdcValues[j_general][1],
      AdcValues[j_general][2], AdcValues[j_general][3], AdcValues[j_general][4],
      AdcValues[j_general][5], AdcValues[j_general][6], AdcValues[j_general][7],
      tiempo[j_general][0], tiempo[j_general][1]);
   //Guardamos en el archivo la muestra obtenida de cada canal con su estampa de
   tiempo en segundos y microsegundos
64   }
65 }
66 fclose(file); //Cerramos el descriptor de archivo
67 flag_triggered=0; //Bajamos la bandera flag_triggered
68 i_parcial=-1; //Reiniciamos el contador parcial
69 i_general=0; //Reiniciamos el contador general
70 MainLoop++; //Incrementamos el contador del bucle principal
/*This loop proves that you can close and re-init peacefully the librairie. Prove
   it several times (e.g. 3) and then finish the code.*/
71 //if (MainLoop == 1000 //Descomentar estas dos líneas si desea que el bucle se
   ejecute una determinada cantidad de veces en lugar de de forma indefinida
72 //    break;
73 }
74 close(fifo_d); //Cerramos el descriptor de archivo de la FIFO
75 printf("ADC_DAC_Close\r\n");

```

```

76     int CloseCode = ADC_DAC_Close(); //Finalizamos la comunicación SPI y ponemos al
77     conversor en standby
78     {
79         error = -7;           //Si la función anterior falla, retornamos el valor -7
80     }
81     printf("Test ADDA finished with returned code %d and error %d\r\n", RetCode,
82 error);
83     return error;          //Retorno de la variable de control error
84 }
```

Funcionamiento:

Puesto que los bucles principales representan la estructura más importante del programa, se explicará brevemente su funcionamiento:

- **Bucle secundario:**

Como se mencionó anteriormente, el bucle secundario es el encargado de realizar las mediciones en forma constante. Previo a ello, se evalúa el estado de la variable bandera **flag_signal**, ya que si la misma posee el valor 1 significa que ha arribado una señal desde el programa de acceso remoto y se deben volver a aplicar los parámetros de configuración (línea 9 a línea 19).

El paso siguiente es obtener el tiempo actual y almacenarlo en el arreglo tiempo (línea 21, 22 y 23). Estas son las estampas de tiempo en segundos más microsegundos de cada muestra. Luego se llama a la función lectura (línea 24) en la cual se obtienen las muestras de los 8 canales y se almacenan en **Adc_Values**.

El primer punto de evaluación es determinar si la muestra obtenida en el canal seleccionado como canal disparador por el usuario ha superado el nivel de disparo elegido (línea 25). De ser este el caso, se debe desencadenar el guardado de las muestras en el archivo, sin embargo, primero deben obtenerse las muestras post disparo especificadas por el usuario. Por ello es levantada la variable bandera **flag_triggered**.

Si esta bandera se encuentra en valor 1 comienza a correr el contador **i_parcial**, el cual al llegar a la cantidad de muestras post disparo produce el salto de bucle (línea 35). Al salir del bucle de medición ya no se obtienen nuevas muestras y se prosigue con el almacenamiento de las muestras ya tomadas en un nuevo archivo.

Por otro lado, si todavía no se alcanzan la cantidad de muestras post disparo o si no se ha producido el disparo propiamente dicho, las muestras son constantemente almacenadas en el arreglo **AdcValues**. Para recorrer este arreglo y así almacenar una nueva muestra en la fila siguiente se utiliza el contador **i_general**. Este arreglo que actúa como buffer de almacenamiento posee limitaciones en cuanto a la cantidad de muestras que puede almacenar sin llenar la memoria RAM. En base a pruebas se determinó que la cantidad máxima de muestras es de aproximadamente 40000, lo que equivale, a una frecuencia de muestreo de 390

SPS, a cerca de 102 segundos de muestreo. Para evitar la saturación de la RAM se colocó un límite máximo de 39000 (de 0 a 38999) muestras; se tomó un margen de seguridad. Esta evaluación se realiza en la línea 37 y, de ser el caso, se reinicia el contador **i_general** a 0. De esta forma las nuevas muestras comenzarán a sobrescribir a las muestras más antiguas almacenadas en las primeras posiciones del arreglo.

Si este límite de 39000 muestras no es alcanzado simplemente se incrementa el contador general.

Lo anteriormente explicado se ilustra con un ejemplo en la figura siguiente:

Fila	Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) +	uS
1	136926	235239	317421	381452	428345	458850	474805	242561	1626963600	910014
2	137362	235878	318008	382071	429014	459037	475420	242346	1626963600	912587
3	137409	236010	318166	382229	429035	459083	475503	241308	1626963600	915154
4	137281	235872	318074	382085	429080	459089	475303	242378	1626963600	917722
5	137257	235820	318019	382011	428825	458862	475174	242655	1626963600	920290
6	136969	235497	317695	381608	428467	458466	474834	243092	1626963600	922858
7	137178	235628	317688	381696	428602	458645	474941	242701	1626963600	925426
8	137169	235675	317894	381786	428781	458863	475168	244874	1626963600	927994
9	137341	235972	318268	382153	428997	459315	475763	242497	1626963600	930562
10	137640	236469	318695	382735	429650	459847	476172	241921	1626963600	933130
11	137491	236267	318643	382751	429628	459773	476055	242092	1626963600	935699
12	137424	236108	318443	382527	429410	459525	475884	242156	1626963600	938267
13	137325	235867	318328	382322	429204	459239	475480	242674	1626963600	940836
14	137041	235696	317914	381975	428950	458954	475310	243374	1626963600	943405
15	137274	235816	318046	382019	428985	459028	475386	242539	1626963600	945974
16	137294	235936	318155	382219	429114	459120	475667	4907537	1626963600	948542
17	137508	236163	318353	382401	429443	459701	476213	4907578	1626963600	951111
18	137727	236555	318868	382829	429769	459897	476332	4907633	1626963600	953679
19	137638	236406	318708	382950	429809	459816	476130	4907607	1626963600	956247
20	137395	236063	318350	382504	429590	459683	476184	4907645	1626963600	958822
21	137318	235960	318380	382358	429265	459352	475599	4907602	1626963600	961390
22	137129	235658	318043	382036	428971	459150	475568	4907694	1626963600	963958
23	137237	235896	318159	382166	429194	459181	475596	4907567	1626963600	966526
24	137313	236092	318290	382448	429189	459380	475824	4907631	1626963600	969094
25	137434	236266	318583	382684	429688	459945	476468	4907627	1626963600	971662
26	141239	237560	318802	382505	429273	459355	475592	4907541	1626963600	974230
27	137800	236655	318704	382528	429364	459234	475580	4907612	1626963600	976855
28	137457	236105	318303	382332	429194	459123	475501	4907650	1626963600	979424
29	137333	235857	318084	382024	428830	458882	475069	4907682	1626963600	981993
30	137116	235816	317827	381894	428797	458801	475103	4907643	1626963600	984565
31	137214	235811	317925	382000	428862	458894	475254	4907649	1626963600	987133
32	137305	235875	318097	382204	429139	462182	476710	4907671	1626963600	989701
33	137155	235715	318098	382667	430095	460708	477252	4907709	1626963600	992325
34	137552	236257	318727	383095	430268	460518	476861	4907542	1626963600	994893
35	137433	236417	318740	382963	429919	460046	476414	4907657	1626963600	997461
36	137354	236064	318405	382580	429677	459907	476328	4907676	1626963601	257905
37	137291	235936	318282	382554	429535	459653	475901	4907546	1626963601	260474

Figura 38: Proceso de disparo y guardado de muestras.

En este ejemplo se han configurado los siguientes parámetros:

- Canal de disparo 7.
- Nivel de disparo de 2.5 V.
- Cantidad de muestras pre disparo igual a 15.
- Cantidad de muestras post disparo igual a 20.

En la muestra número 16 tomada por el canal 7 se supera el nivel de disparo:

$$4907537 \longrightarrow \frac{4907537 * 5}{2^{23}} = 2.93 V$$

Por lo tanto, se toman las 15 muestras antes del disparo, la muestra que produjo el disparo y 20 muestras más a partir de este.

- **Bucle primario:**

El bucle primario la primera acción que realiza es la declaración de los arreglos de almacenamiento de muestras y tiempo **tiempo** y **AdcValues** (líneas 3 y 4). Declarar los arreglos dentro de este bucle permite que estos se reinicien cada vez que se vuelve a entrar en el bucle secundario. El fin de esto es evitar que en los mismos queden muestras tomadas en instantes previos y ya almacenadas en un archivo. No reiniciar estos arreglos podría traer como consecuencia tomar muestras pre disparo que no corresponden con el periodo actual de medición.

La siguiente tarea que realiza es llamar a la función **archivo()**, la cual recuerde es la encargada de generar el nuevo archivo donde se almacenarán las muestras.

Puesto que se deben escribir en el archivo desde la primera muestra pre disparo hasta la última muestra post disparo, se toma la primera como referencia. Para ello se utiliza el valor remanente de **i_general**. De esta forma, al número de la última muestra se le restan la suma de la cantidad de muestras pre disparo y post disparo que es el total de muestras a almacenar (línea 47).

Posterior a esto se presentan dos posibles situaciones:

1. Que el grupo de muestras a almacenar se encuentre en posiciones intermedias del arreglo.
2. Que el grupo de muestras a almacenar se encuentre entre las últimas y las primeras posiciones del arreglo.

La primera situación no representa mayores inconvenientes ya que lo único que se debe hacer es guardar desde la muestra número **i_referencia** hasta la muestra número **i_referencia+N_total+1** (más 1 para tomar la muestra que produjo el disparo). Esto se hace en el bucle for de la línea 61, imprimiendo en el archivo cada muestra de cada canal en el formato mostrado.

La segunda situación se ilustra en la figura 5*. En este caso no podemos simplemente almacenar desde la muestra **i_referencia** ya que esta cuenta es negativa:

- **i_general** = 25
- Muestras pre disparo = 10
- Muestras post disparo = 25
- **N_total** = 10 + 25 = 35

$$i_{referencia} = i_{general} - N_{total} = -10$$

Por esta razón se decidió repartir la escritura de las muestras en dos bucles for. El primero debe almacenar desde la muestra número **38999+i_referencia = 38989** hasta la muestra **38999**. El segundo debe escribir desde la muestra número **0** hasta la muestra número **i_general = 25**.

Por último, se debe cerrar el archivo y reiniciar los contadores y banderas utilizadas (líneas 66 a 69).

Fila	Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) +	uS
38989	136926	235239	317421	381452	428345	458350	474805	242561	1626963600	910014
38990	137362	235878	318008	382071	429014	459037	475420	242346	1626963600	912587
38991	137409	236010	318166	382229	429035	459083	475503	241308	1626963600	915154
38992	137281	235872	318074	382085	429080	459039	475303	242378	1626963600	917722
38993	137257	235820	318019	382011	428825	458862	475174	242655	1626963600	920290
38994	136969	235497	317695	381608	428467	458466	474834	243092	1626963600	922858
38995	137178	235628	317688	381696	428602	458645	474941	242701	1626963600	925426
38996	137169	235675	317894	381786	428781	458863	475163	244874	1626963600	927994
38997	137341	235972	318268	382153	428997	459315	475763	242497	1626963600	930562
38998	137640	236469	318695	382735	429650	459847	476172	241921	1626963600	933130
38999	137491	236267	318643	382751	429628	459773	476055	242092	1626963600	935699
0	137424	236108	318443	382527	429410	459525	475884	242156	1626963600	938267
1	137325	235867	318328	382322	429204	459239	475480	242674	1626963600	940836
2	137041	235696	317914	381975	428950	458954	475310	243374	1626963600	943405
3	137274	235816	318046	382019	428985	459028	475386	242539	1626963600	945974
4	137294	235936	318155	382219	429114	459120	475667	4907537	1626963600	948542
5	137508	236163	318353	382401	429443	459701	476213	4907578	1626963600	951111
6	137727	236555	318868	382829	429769	459897	476332	4907633	1626963600	953679
7	137638	236406	318708	382950	429809	459816	476130	4907607	1626963600	956247
8	137395	236063	318350	382504	429590	459683	476184	4907645	1626963600	958822
9	137318	235960	318380	382358	429265	459352	475599	4907602	1626963600	961390
10	137129	235658	318043	382036	428971	459150	475568	4907694	1626963600	963958
11	137237	235896	318159	382166	429194	459181	475596	4907567	1626963600	966526
12	137313	236092	318290	382448	429189	459380	475824	4907631	1626963600	969094
13	137434	236266	318583	382684	429688	459945	476468	4907627	1626963600	971662
14	141239	237560	318802	382505	429273	459355	475592	4907541	1626963600	974230
15	137800	236655	318704	382528	429364	459234	475580	4907612	1626963600	976855
16	137457	236105	318303	382332	429194	459123	475501	4907650	1626963600	979424
17	137333	235857	318084	382024	428830	458882	475069	4907682	1626963600	981993
18	137116	235816	317827	381894	428797	458801	475103	4907643	1626963600	984565
19	137214	235811	317925	382000	428862	458894	475254	4907649	1626963600	987133
20	137305	235875	318097	382204	429139	462182	476710	4907671	1626963600	989701
21	137155	235715	318098	382667	430095	460708	477252	4907709	1626963600	992325
22	137552	236257	318727	383095	430268	460518	476861	4907542	1626963600	994893
23	137433	236417	318740	382963	429919	460046	476414	4907657	1626963600	997461
24	137354	236064	318405	382580	429677	459907	476328	4907676	1626963601	257905
25	137291	235936	318282	382554	429585	459653	475901	4907546	1626963601	260474

Figura 39: Proceso de disparo y guardado de muestras entre las últimas y las primeras muestras del buffer .

Como resultado se obtienen archivos cuyo nombre posee el siguiente formato:

d|m|y-H:M:S

Por ejemplo: “0015-29|12|21-19:43:54.txt”. Y su encabezado y contenido:

```

Wed Dec 29 19:43:54 2021 Parámetros: Canal de disparo: 1 Frecuencia de muestreo: 30000 Nivel de disparo= 1.500000 Cantidad de muestras post trigger = 500 Cantidad de muestras pre trigger = 300 Número de archivo = 15Volts/cuenta = 5000000/8388608 Canal 0 (uV) Canal 1 (uS)
(uV) Canal 2 (uV) Canal 3 (uV) Canal 4 (uV) Canal 5 (uV) Canal 6 (uV) Canal 7 (uV) Tiempo: canal 0 (S) +
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
832614 847677 862886 877447 892226 906783 922021 938248 1634741529 124592
953393 969037 984859 1000830 1016420 1032779 1049411 1066684 1634741529 127208
1085099 1102879 1120202 1137461 1155041 1172410 1189801 1208170 1634741529 129830
1225322 1243005 1261040 1278634 1296227 1313563 1331614 1350188 1634741529 132460
1367667 1385324 1403622 1422291 1441035 1459607 1478573 1499416 1634741529 135051
1519315 1539355 1559987 1579621 1598739 1618177 1638444 1659028 1634741529 137683
1678993 1699629 1720506 1741522 1762194 1782079 1802074 1823208 1634741529 140284
1843338 1863530 1884241 1904617 1924334 1944174 1964490 1985684 1634741529 142957
2005789 2025728 2046384 2066537 2086711 2107103 2127402 2148816 1634741529 145557
2169009 2189696 2211146 2232452 2253774 2274842 2296045 2318346 1634741529 148125
2339663 2361208 2383551 2405094 2426776 2448742 2470466 2493474 1634741529 150693
2515069 2536998 2559469 2581169 2602698 2623782 2644444 2666311 1634741529 153298
2689381 2712032 2733638 2754879 2775975 2798054 2819921 2843088 1634741529 156015
2867229 2889157 2910647 2931732 2952296 2972616 2993181 3014776 1634741529 158672
3036566 3057008 3077750 3098698 3119124 3139376 3160254 3181228 1634741529 161321
3205515 3225717 3245669 3265581 3285510 3305295 3325251 3345567 1634741529 164015
3364978 3385932 3406173 3426700 3446782 3467074 3487035 3508123 1634741529 166648
3529353 3549660 3568846 3587298 3605445 3624236 3642776 3661893 1634741529 169388
3680034 3697814 3715796 3733408 3751190 3768668 3786260 3804678 1634741529 171991

```

Figura 40: Archivo de texto con las muestras almacenadas.

El cual, al ser exportado a un archivo de Excel queda:

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10
Wed Dec 29 19:43:54 2021									
Parámetros:									
Canal de disparo: 1									
Frecuencia de muestreo: 30000									
Nivel de disparo= 1.500000									
Cantidad de muestras post trigger = 500									
Cantidad de muestras pre trigger = 300									
Número de archivo = 15Volts/cuenta = 5000000/8388608									
Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) + uS	
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
832614	847677	862886	877447	892226	906783	922021	938248	1634741529	124592
953393	969037	984859	1000830	1016420	1032779	1049411	1066684	1634741529	127208
1085099	1102879	1120202	1137461	1155041	1172410	1189801	1208170	1634741529	129830
1225322	1243005	1261040	1278634	1296227	1313563	1331614	1350188	1634741529	132460
1367667	1385324	1403622	1422291	1441035	1459607	1478573	1499416	1634741529	135051
1519315	1539355	1559987	1579621	1598739	1618177	1638444	1659028	1634741529	137683
1678993	1699629	1720506	1741522	1762194	1782079	1802074	1823208	1634741529	140284
1843338	1863530	1884241	1904617	1924334	1944174	1964490	1985684	1634741529	142957
2005789	2025728	2046384	2066537	2086711	2107103	2127402	2148816	1634741529	145557
2169009	2189696	2211146	2232452	2253774	2274842	2296045	2318346	1634741529	148125
2339663	2361208	2383551	2405094	2426776	2448742	2470466	2493474	1634741529	150693
2515069	2536998	2559469	2581169	2602698	2623782	2644444	2666311	1634741529	153298
2689381	2712032	2733638	2754879	2775975	2798054	2819921	2843088	1634741529	156015
2867229	2889157	2910647	2931732	2952296	2972616	2993181	3014776	1634741529	158672
3036566	3057008	3077750	3098698	3119124	3139376	3160254	3181228	1634741529	161321
3205515	3225717	3245669	3265581	3285510	3305295	3325251	3345567	1634741529	164015
3364978	3385932	3406173	3426700	3446782	3467074	3487035	3508123	1634741529	166648

Figura 41: Archivo Excel con las muestras almacenadas.

Notar que en caso de no poseer suficientes muestras pre disparo al momento de generar el archivo, los espacios vacíos son rellenados con ceros. Esta situación puede darse cuando el disparo se produzca al poco tiempo de haber iniciado el dispositivo o justo después de haber generado un archivo y reiniciado los arreglos de almacenamiento.

Listado de errores:

- **error = 0** : El programa transcurrió sin errores.
- **error = -1** : No pudo abrirse el archivo Parametros.txt en la función parametros().
- **error = -2** : No pudo abrirse el archivo Parametros_respaldo.txt en la función parametros().
- **error = -3** : No pudo abrirse el archivo lastfile.txt en modo lectura en la función archivo().
- **error = -4** : No pudo abrirse el archivo lastfile.txt en modo escritura en la función archivo().
- **error = -5** : No pudo iniciarse la comunicación serie con la placa.
- **error = -6** : Init es distinto de 0, no pudo iniciarse el ADC.
- **error = -7** : No pudo cerrarse la comunicación serie con la placa y colocarla en standby.
- **RetCode = -1** : No se encontró y por lo tanto no se eliminó ningún archivo con el nombre especificado en la función archivo().
- **RetCode = -2** : No pudo crearse la FIFO, porque ya existe u ocurrió algún error.
- **RetCode = -3**: Error al abrir la FIFO.
- **RetCode = -4** : Error al escribir en la FIFO.
- **RetCode = -5** : No pudo reiniciarse el ADC en el bucle secundario del main.

6 INSTALACIÓN DEL MÓDULO RTC

Anteriormente se vio como conectar el módulo RTC a la placa conversora. Sin embargo, para que este módulo funcione correctamente debe instalarse y configurarse.

El módulo se conecta a través de la interfaz I2C.

Los pasos son los siguientes:

1. Actualizar:

Antes de empezar la instalación, debe asegurarse de que todos los paquetes estén actualizados, por lo que, primero se debe realizar una actualización y luego instalar el software I2C:

```
sudo apt-get update && sudo apt-get upgrade - yes
```

```
sudo apt-get install i2c-tools
```

```
pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get upgrade - yes
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.7 kB]
Get:3 http://archive.raspberrypi.org/debian buster/main armhf Packages [393 kB]
Reading package lists... Done
E: Repository 'http://raspbian.raspberrypi.org/raspbian buster' changed its 'Suite' value from 'stable' to 'oldstable'
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.
pi@raspberrypi:~ $ sudo apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (4.1-1).
i2c-tools set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 113 not upgraded.
```

Figura 42: Instalación de las herramientas I2C.

También debe activar el bus I2C desde la configuración de la Raspberry Pi:

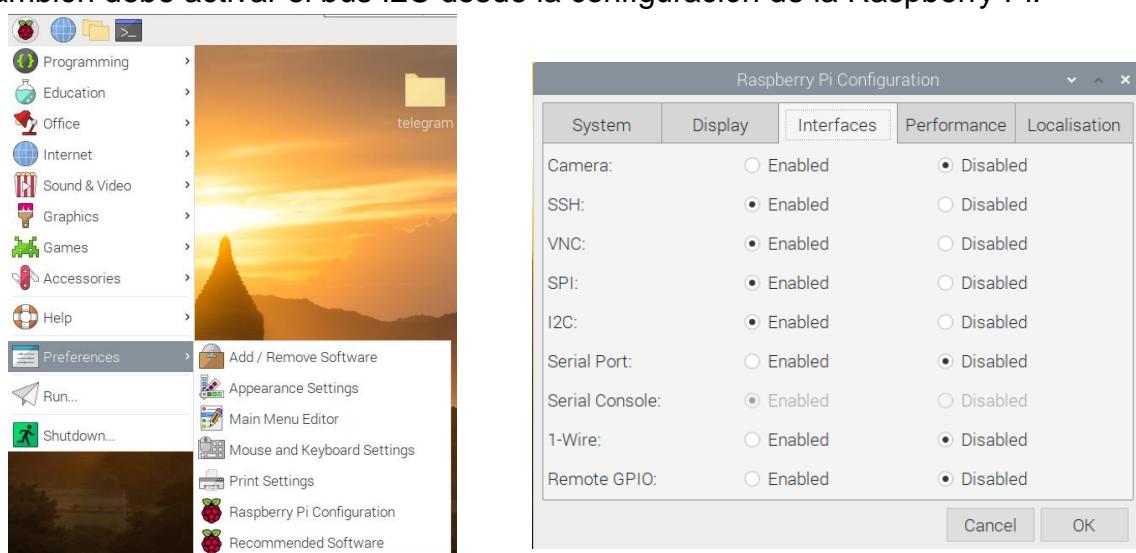
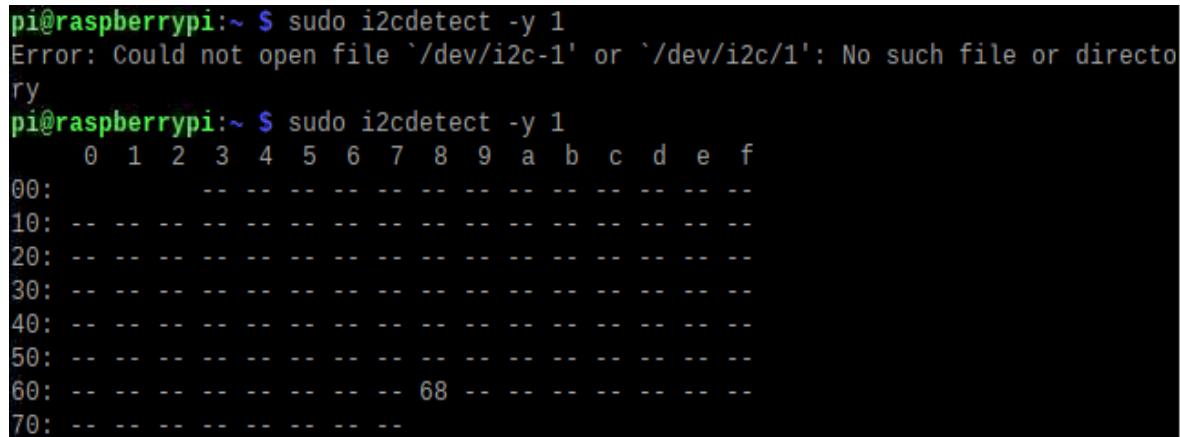


Figura 43: Habilitación de la comunicación I2C.

2. Detectar:

Lo siguiente es determinar si la placa detecta al módulo. Para eso utilice el comando:

```
sudo i2cdetect -y 1
```



```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
Error: Could not open file '/dev/i2c-1' or '/dev/i2c/1': No such file or directory
pi@raspberrypi:~ $ sudo i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          --  --  --  --  --  --  --  --  --  --  --  --
10:          --  --  --  --  --  --  --  --  --  --  --  --
20:          --  --  --  --  --  --  --  --  --  --  --  --
30:          --  --  --  --  --  --  --  --  --  --  --  --
40:          --  --  --  --  --  --  --  --  --  --  --  --
50:          --  --  --  --  --  --  --  --  --  --  --  --
60:          --  --  --  --  --  --  --  --  --  68  --  --
70:          --  --  --  --  --  --  --  --  --  --  --  --
```

Figura 44: Detección del módulo.

Si el resultado es el ID #68 significa que el módulo RTC funciona ya que es la dirección normalmente asignada al RTC DS3231 y también del DS1307.

Se puede agregar un soporte para el RTC agregando una superposición del árbol de dispositivos. Para ello debemos editar el archivo config.txt.

```
sudo nano /boot/config.txt
```

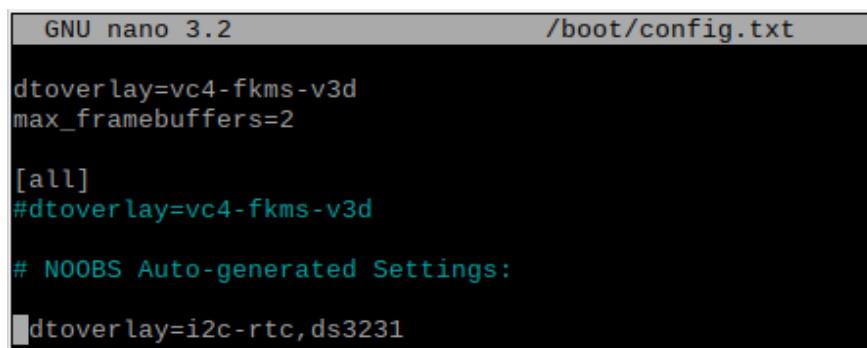
Para editar la configuración de la placa y agregar aquel que coincida con su chip RTC, escriba:

```
dtoverlay=i2c-rtc,ds1307
```

O

```
dtoverlay=i2c-rtc,ds3231
```

al final del archivo.



```
GNU nano 3.2                               /boot/config.txt

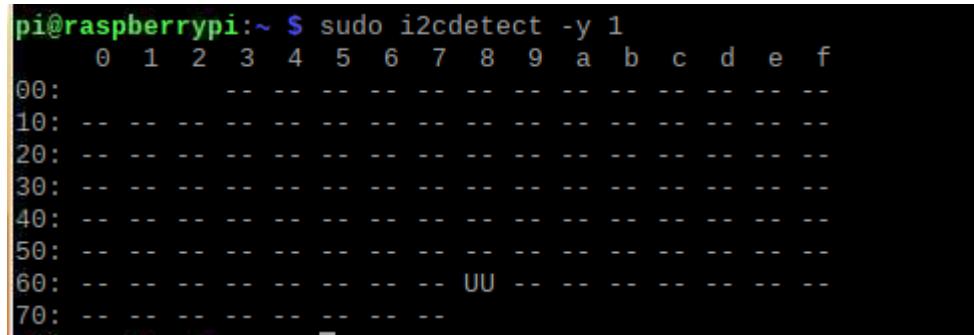
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d

# NOOBS Auto-generated Settings:
dtoverlay=i2c-rtc,ds3231
```

Figura 45: Modificación del archivo config.txt.

Guarde y ejecute **sudo reboot** para reiniciar la placa y luego ejecute **sudo i2cdetect -y 1** deberá verse UU donde antes estaba la dirección 0x68.



```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --
10: --
20: --
30: --
40: --
50: --
60: -- UU --
70: --
```

Figura 46: Nueva detección del módulo RTC.

3. Cargar el módulo en el kernel:

Antes de avanzar es importante aclarar que el módulo RTC DS3231 y el DS1307 son compatibles y sustitutos, por lo tanto, se puede instalar el DS3231 con los mismos comandos del DS1307 e incluso colocando el nombre ds1307 en la sintaxis usada.

En base a lo dicho, ejecute **sudo modprobe rtc-ds1307**.

En súper usuario: **sudo bash**

Y luego:

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

o

```
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Luego salga del super usuario:

```
exit
```

Puede comprobar los módulos cargados en el kernel en la ruta */lib/modules/(sub directorio)* abriendo el archivo *modules.alias* verá los alias de los distintos módulos cargados:

```

modules.alias - Mousepad
File Edit Search View Document Help
alias i2c:m41t11 rtc_ds1307
alias i2c:m41t00 rtc_ds1307
alias i2c:m41t0 rtc_ds1307
alias i2c:ds3231 rtc_ds1307
alias i2c:ds1341 rtc_ds1307
alias i2c:ds1340 rtc_ds1307
alias i2c:ds1388 rtc_ds1307
alias i2c:ds1339 rtc_ds1307
alias i2c:ds1338 rtc_ds1307
alias i2c:ds1337 rtc_ds1307
alias i2c:ds1308 rtc_ds1307
alias i2c:ds1307 rtc_ds1307
alias of:N*T*Cepson,rx8130C* rtc_ds1307
alias of:N*T*Cepson,rx8130 rtc_ds1307
alias of:N*T*Cisil,isl12057C* rtc_ds1307
alias of:N*T*Cisil,isl12057 rtc_ds1307
alias of:N*T*Cepson,rx8025C* rtc_ds1307
alias of:N*T*Cepson,rx8025 rtc_ds1307
alias of:N*T*Cpericom,pt7c4338C* rtc_ds1307
alias of:N*T*Cpericom,pt7c4338 rtc_ds1307
alias of:N*T*Cmicrochip,mcp7941xC* rtc_ds1307
alias of:N*T*Cmicrochip,mcp7941x rtc_ds1307
alias of:N*T*Cmicrochip,mcp7940xC* rtc_ds1307
alias of:N*T*Cmicrochip,mcp7940x rtc_ds1307
alias of:N*T*Cst,m41t11C* rtc_ds1307
alias of:N*T*Cst,m41t11 rtc_ds1307
alias of:N*T*Cst,m41t00C* rtc_ds1307
alias of:N*T*Cst,m41t00 rtc_ds1307
alias of:N*T*Cst,m41t0C* rtc_ds1307
alias of:N*T*Cst,m41t0 rtc_ds1307
alias of:N*T*Cmaxim,ds3231C* rtc_ds1307
alias of:N*T*Cmaxim,ds3231 rtc_ds1307
alias of:N*T*Cdallas,ds1341C* rtc_ds1307

```

Find: 3231 Next Previous Highlight All Match Case

Figura 47: Archivo modules.alias con los módulos cargados.

Ya se encuentra cargado el módulo al kernel y cada vez que la Raspberry se inicie podrá comprobar la hora en el RTC:

sudo hwclock -r

```

pi@raspberrypi:~ $ sudo hwclock -r
2021-09-21 20:17:32.807871-03:00

```

Figura 48: Visualización de la fecha y hora de la placa.

Si es la primera vez que lo ejecuta y no posee conectividad a internet la Raspberry Pi devolverá la fecha 1 de enero de 2000. Para configurar la hora, la manera más sencilla es conectarla a internet para que automáticamente se actualice.

Si no, de forma manual, primero deberá configurar la hora del sistema y luego escribirla en el RTC. Esto se puede hacer con:

```
sudo hwclock --set --date = "$(date" +%m /%d /%Y %H:%M:%S ")"
```

o con

```
sudo date 08050822.
```

Donde:

- 08 es el mes
- 05 es el día
- 08 es la hora
- 22 son los minutos

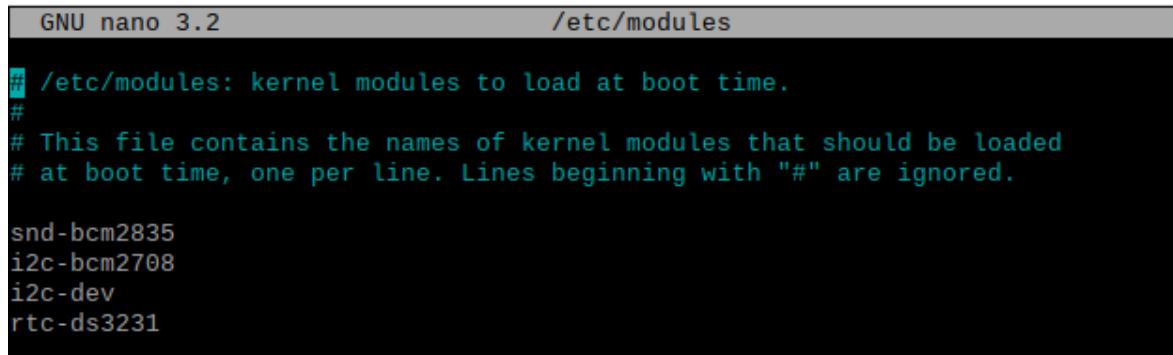
Y luego **sudo hwclock -w**.

Lograr que la Raspberry obtenga la hora del RTC:

Ya se ha conseguido que la Raspberry Pi configure la hora en el RTC, sin embargo, se le debe indicar que obtenga la hora del RTC cada vez que se inicie. Para ello hay que modificar el archivo *modules* en el directorio etc.

Este archivo contiene los nombres de los módulos del kernel que se deben cargar en el momento del arranque.

En el mismo se deben agregar las siguientes líneas al final del archivo:



```
GNU nano 3.2          /etc/modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

snd-bcm2835
i2c-bcm2708
i2c-dev
rtc-ds3231
```

Figura 49: Modificación del archivo modules.

El siguiente paso es añadir el dispositivo DS3231 como reloj del sistema en el archivo rc.local. Para ello ejecutar:

```
sudo nano /etc/rc.local
```

Y añadir las siguientes líneas justo antes de «exit 0»

```
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

```
sudo hwclock -s
```

```
date
```

Quedando:

```
GNU nano 3.2                               /etc/rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sudo hwclock -s
date
|
exit 0
```

Figura 50: Modificación del archivo rc.local.

Desconecte la conexión a internet y reinicie la Raspberry Pi para probar el correcto funcionamiento del módulo.

Efectivamente se comprobó que al iniciar la Raspberry Pi 10 minutos después de haberla apagado, y sin conexión a internet, la hora mostrada era la correcta. Al desconectar el módulo y reiniciar la placa sin internet la hora mostrada era la antigua, lo que indica que efectivamente la Raspberry Pi estaba obteniendo la hora del módulo RTC.

Consideración: al conectar la placa Raspberry PI a internet, luego de aproximadamente 1 minuto, es actualizada la hora a la actual. Este retardo en la actualización de la hora cuando no posee el módulo RTC debe tenerse en cuenta, ya que si, durante este intervalo de tiempo se ejecuta el programa de medición y se produce un disparo, el archivo generado poseerá una hora que no es la correcta.

7 INTERFAZ DE USUARIO

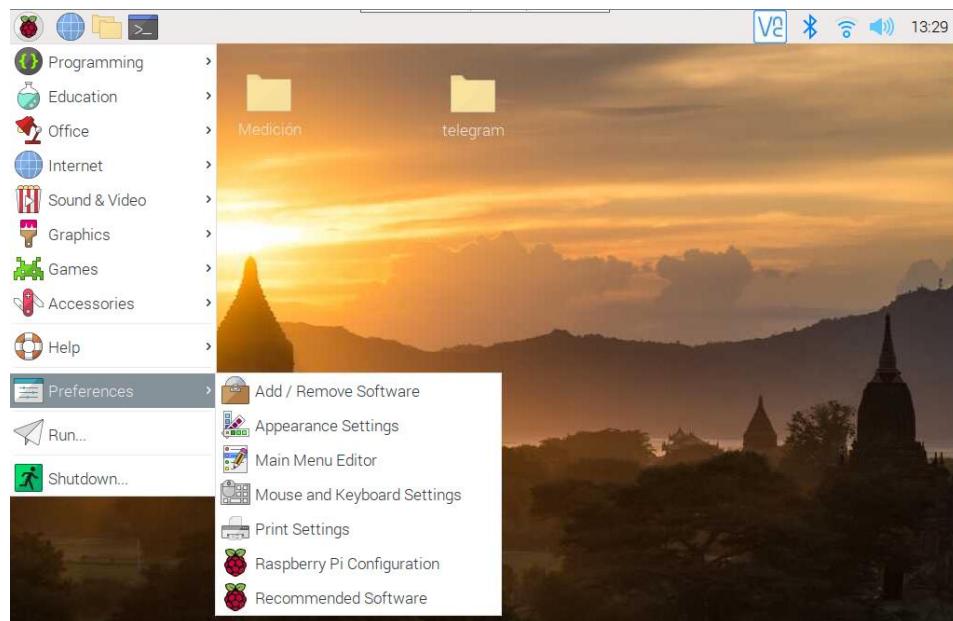
El usuario posee diversas formas de acceder a los parámetros y archivos generados. La más directa de todas es conectando el dispositivo a un monitor, mouse y teclado mediante sus puertos. Sin embargo, también existen varios métodos de acceso remoto a la placa. Se hará una división entre el acceso remoto dentro de la misma red de área local y otra fuera de esta.

7.1 ACCESO REMOTO DENTRO DE LA RED DE ÁREA LOCAL

Todos los métodos que se explicarán requieren conocer la dirección IP y el puerto de la placa Raspberry Pi antes de realizar cualquier conexión.

Para ello se recomienda el programa Nmap, el cual escanea todas las direcciones IP dentro de la red a través del envío de un ping y solo aquellas direcciones que respondan serán las que aparecerán en el listado. Para esto se ejecuta nmap desde la terminal de comandos a través del comando nmap 192.168.0.0/24 (averigüe cuál es su red previo a realizar esta acción). Con esto se le indica desde qué dirección IP dentro de la red desea que escanee. Cuando finalice arrojará un listado de las IP ocupadas e información relacionada como por ejemplo el puerto que está utilizando y la dirección MAC de cada dispositivo. Es gracias a esta dirección MAC que reconocerá la placa Raspberry Pi pues contiene el nombre de la empresa que fabrica el dispositivo, en este caso Raspberry Pi Foundation.

Las placas Raspberry Pi tienen por defecto desactivadas las conexiones SSH, por lo que no aparecerá un puerto asociado lo que ocasionará problemas al intentar conectarse remotamente. Por lo tanto, debe habilitarse. Esto se hace conectando la placa a un monitor y desde configuraciones de red habilitar dichas conexiones:



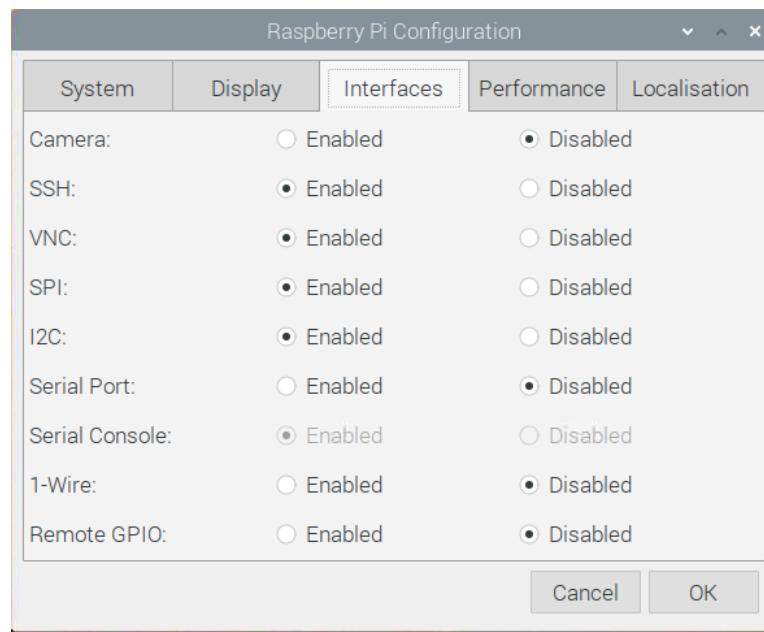


Figura 51: Habilitación de la comunicación SSH.

```
C:\Users\Administrador>nmap 192.168.0.0/24
Starting Nmap 7.91 < https://nmap.org > at 2022-01-20 13:17 Hora est&ndash;ndar de Argentina
Nmap scan report for 192.168.0.1
Host is up <0.023s latency>.
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 58:D9:D5:28:C7:18 <Tenda Technology,Ltd.Dongguan branch>

Nmap scan report for 192.168.0.100
Host is up <0.020s latency>.
All 1000 scanned ports on 192.168.0.100 are closed
MAC Address: 08:AA:55:7B:9C:A3 <Motorola Mobility, a Lenovo Company>

Nmap scan report for 192.168.0.108
Host is up <0.0066s latency>.
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5900/tcp  open  vnc
MAC Address: B8:27:EB:B3:17 <Raspberry Pi Foundation>

Nmap scan report for 192.168.0.112
Host is up <0.039s latency>.
Not shown: 995 closed ports
PORT      STATE SERVICE
3000/tcp  open  ppp
3001/tcp  open  nessus
7728/tcp  open  interwise
9080/tcp  open  glrp
9998/tcp  open  distinct32
MAC Address: 30:A9:DE:25:50:D2 <LG Innotek>

Nmap scan report for 192.168.0.118
Host is up <0.020s latency>.
All 1000 scanned ports on 192.168.0.118 are closed
MAC Address: 8A:A8:9D:0B:32:10 <Unknown>

Nmap scan report for 192.168.0.110
Host is up <0.00018s latency>.
Not shown: 999 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1027/tcp  open  IIS
1028/tcp  open  unknown
2869/tcp  open  icslap
3580/tcp  open  nati-svrloc
5357/tcp  open  wsdapi
8080/tcp  open  http-proxy

Nmap done: 256 IP addresses (6 hosts up) scanned in 8.07 seconds
```

Figura 52: Escaneo de las direcciones IP ocupadas dentro de la red.

Puede observar que la dirección IP asignada es la 192.168.0.108 y el puerto 22.

Si lo que desea es acceder a la terminal de comandos se recomienda descargar el programa Putty, el cual le permitirá acceder de forma remota a la terminal del dispositivo con su IP y su puerto. Solicitará un usuario y contraseña que por defecto en estas placas el usuario es 'pi' y la contraseña 'raspberry'.

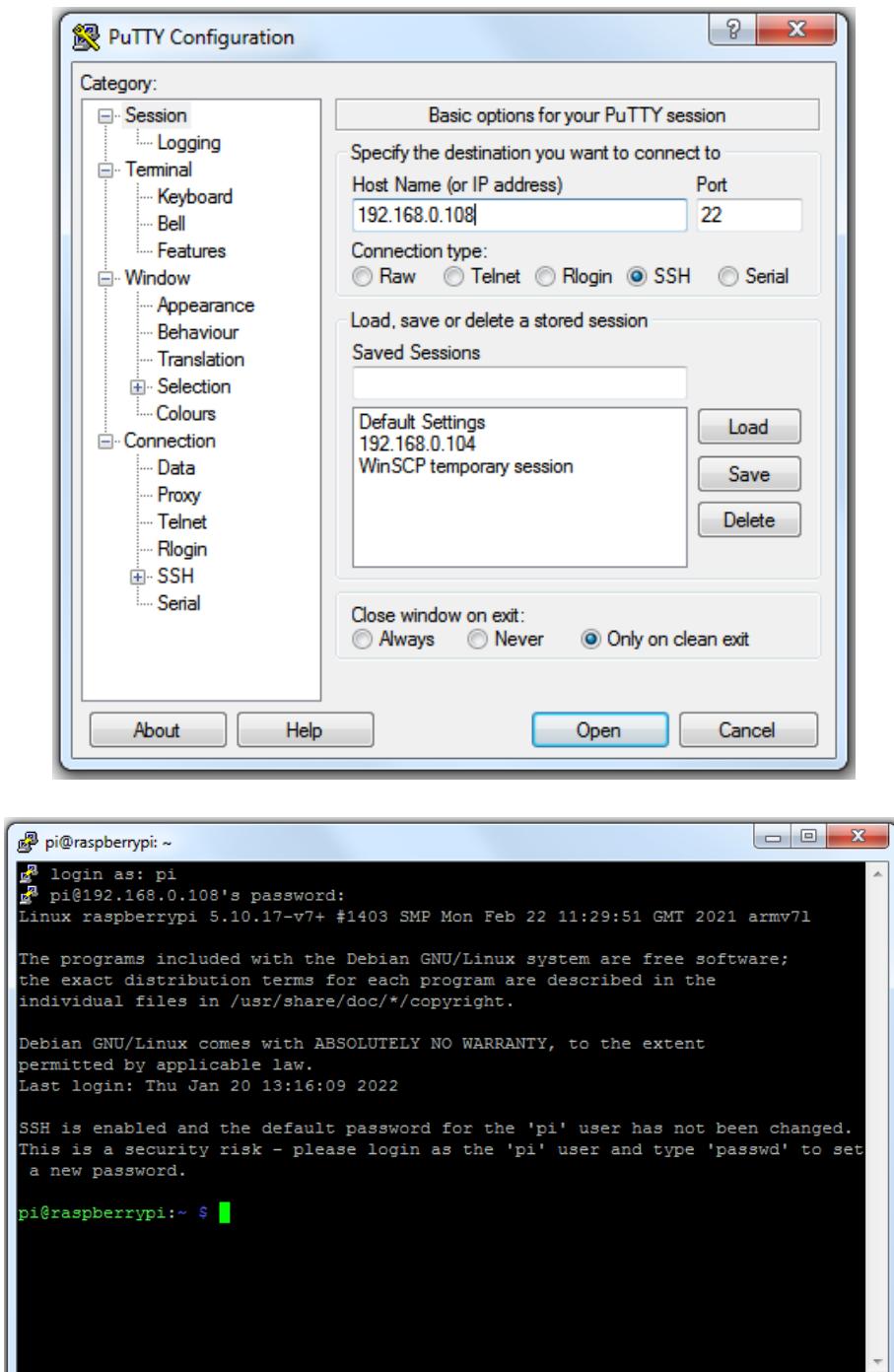


Figura 53: Acceso a la terminal de comandos a través del programa PuTTY.

Para poder acceder a la interfaz gráfica deberá instalar otro programa, tanto en su computadora como en la Raspberry Pi. Desde la terminal de la placa ejecute el comando ***sudo apt-get install -y realvnc-vnc-server realvnc-vnc-viewer*** y una vez instalado ejecute ***vncserver*** lo que le permitirá ejecutar un servidor vnc. Generará un nuevo escritorio (192.168.0.108:1) que usará desde su PC.

```
pi@raspberrypi:~ $ vncserver
VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:34:20)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See https://www.realvnc.com for information on VNC.
For third party acknowledgements see:
https://www.realvnc.com/docs/6/foss.html
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l

VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:44:08)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See https://www.realvnc.com for information on VNC.
For third party acknowledgements see:
https://www.realvnc.com/docs/6/foss.html
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l
On some distributions (in particular Red Hat), you may get a better experience
by running vncserver-virtual in conjunction with the system Xorg server, rather
than the old version built-in to Xvnc. More desktop environments and
applications will likely be compatible. For more information on this alternative
implementation, please see: https://www.realvnc.com/doclink/kb-546

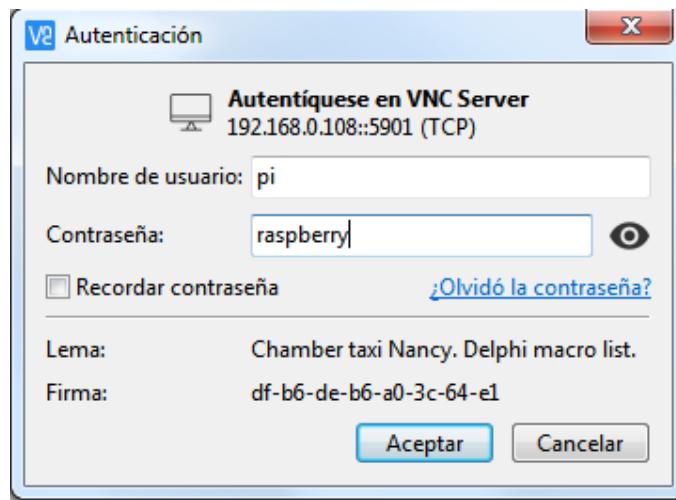
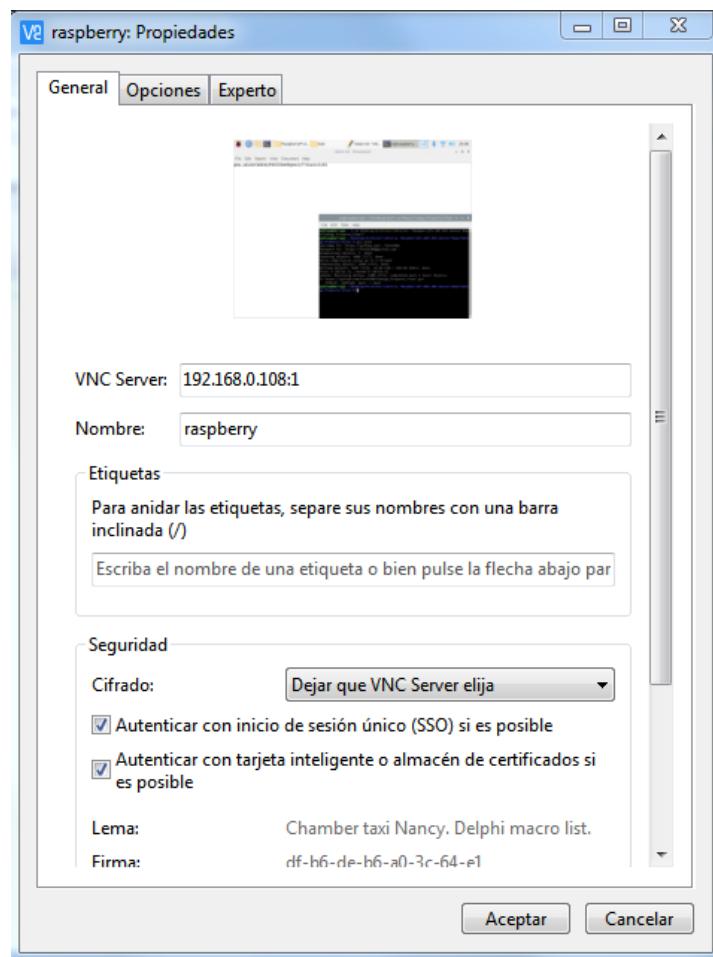
Running applications in /etc/vnc/xstartup

VNC Server catchphrase: "Chamber taxi Nancy. Delphi macro list."
signature: df-b6-de-b6-a0-3c-64-e1

Log file is /home/pi/.vnc/raspberrypi:1.log
New desktop is raspberrypi:1 (192.168.0.108:1)
```

Figura 54: Creación de un servidor VNC.

Descargue el programa VNC Viewer en su computadora y una vez instalado coloque el escritorio generado por el servidor; también pedirá un identificador y luego el usuario y contraseña que son las ya mencionadas anteriormente. De esta forma podrá acceder al escritorio de la placa Raspberry Pi como si la hubiera conectado de forma directa a un monitor, teclado y mouse.



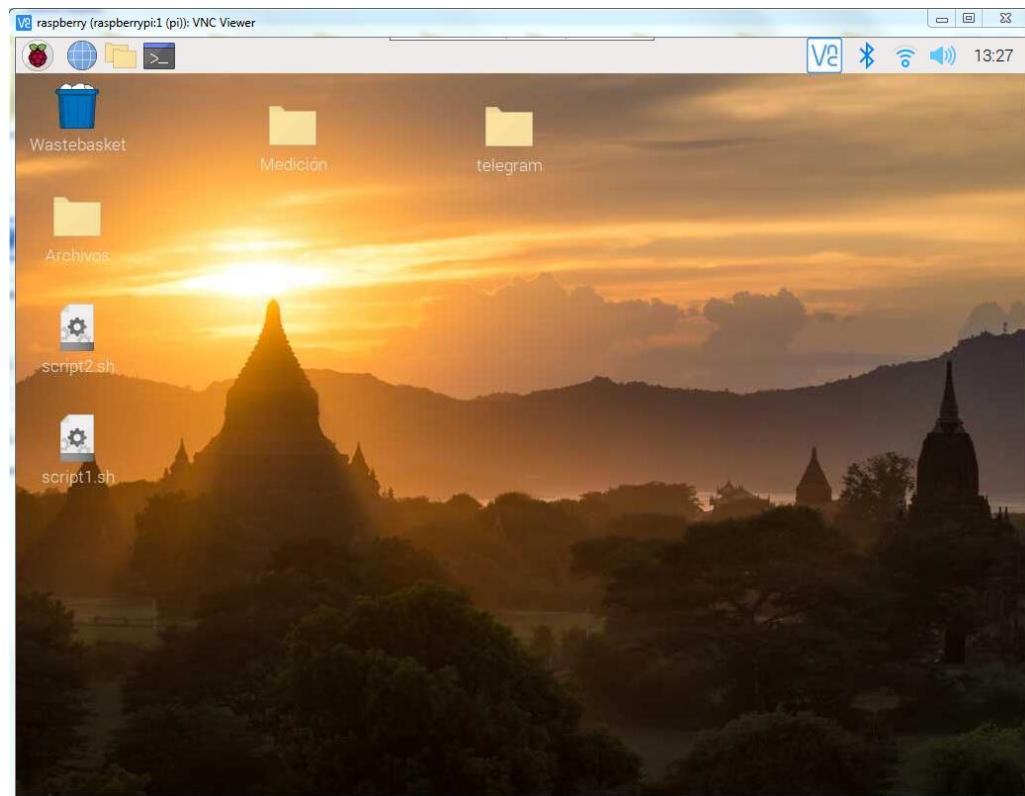
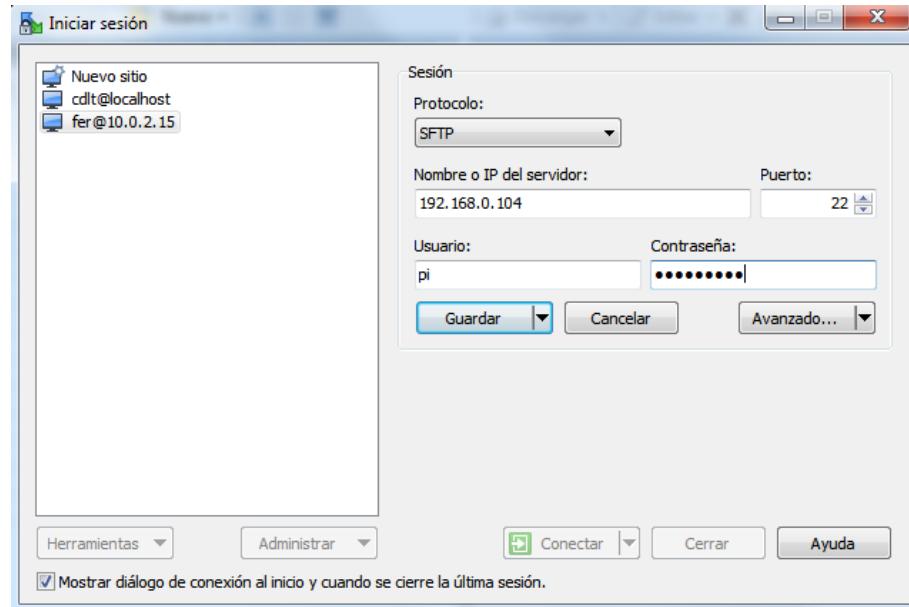


Figura 55: Acceso al escritorio de la Raspberry Pi a través del programa VNC Viewer.

VNC Viewer le permite usar su pantalla como si hubiera conectado el monitor y el mouse de forma directa a la placa, pero no permite el paso de archivos entre la computadora y la Raspberry. Para esto, se recomienda utilizar el programa WinSCP, que permite realizar la transferencia de archivos de una forma muy cómoda utilizando la dirección IP y el puerto asignado a la placa.



Nombre	Tamaño	Modificado	Permisos	Propietario
..		06/04/2021 22:27:49	rwxx-xr-x	pi
libreria		20/04/2021 22:29:17	rwxx-xr-x	pi
build		22/04/2021 10:11:18	rwxx-xr-x	pi
bcm2835-1.68		20/04/2021 22:43:48	rwxx-xr-x	pi
T_po16 8 canales	8 KB	06/04/2021 17:19:08	rw-r--r--	pi
T_po16 3 canales	3 KB	06/04/2021 17:19:10	rw-r--r--	pi
Prueba_16 01.c	32 KB	12/04/2021 22:11:36	rw-r--r--	pi
Prueba_16 00.o	9 KB	20/04/2021 22:44:50	rw-r--r--	pi
Prueba_16 00.c	32 KB	06/04/2021 17:19:04	rw-r--r--	pi
Essentials_C_v1.pdf	4.942 KB	06/04/2021 17:53:18	rw-r--r--	pi
bcm2835-1.68.tar.gz	267 KB	06/04/2021 22:31:55	rw-r--r--	pi
bcm2835.h	95 KB	21/07/2020 17:32:10	rw-r--r--	pi

Figura 56: Transferencia de archivos a través del programa WinSCP.

Observe que se presentan dos pantallas, una con los directorios de su PC y la otra con los de la Raspberry Pi. Simplemente copiando y pegando de una pantalla a otra podrá descargar los archivos generados.

7.2 ACCESO REMOTO FUERA DE LA RED DE ÁREA LOCAL

Puede presentarse la necesidad de descargar los archivos o modificar los parámetros sin que el usuario se encuentre conectado a la misma red que el dispositivo.

Para esto se utilizó el servicio de mensajería de Telegram, y particularmente la interfaz Telegram Bot API. Esta última es una interfaz de programación de Telegram orientada al uso de bots.

Básicamente, los Telegram Bots son cuentas especiales que no requieren un número de teléfono adicional para su creación. Los usuarios pueden interactuar con los bots de dos maneras:

- Enviando mensajes y comandos a los bots abriendo un chat con ellos o agregándolos a grupos.
- Enviando solicitudes directamente desde el campo de entrada escribiendo el @nombre de usuario del bot y una consulta. Esto permite enviar contenido desde bots en línea directamente a cualquier chat, grupo o canal.

Los mensajes, comandos y solicitudes enviados por los usuarios pasan al software que se ejecuta en sus servidores. Un servidor intermediario maneja todo el cifrado y la comunicación con la API de Telegram. Puede comunicarse con este servidor a través de una interfaz HTTPS simple que ofrece una versión simplificada de la API de Telegram. Esa interfaz es la llamada API Bot .

Por lo anteriormente dicho, básicamente se debe crear un bot, el cual será el intermediario entre la interfaz mostrada en un celular o computadora y la placa Raspberry Pi. El usuario puede enviar comandos y mensajes a este bot y este es el encargado de enviársela a la Raspberry Pi, al igual que recibir mensajes de esta.

- **Cómo crear un bot de Telegram.**

En primer lugar, debe tener Telegram instalado con una cuenta habilitada. Lo siguiente es ir a los contactos de Telegram y buscar:

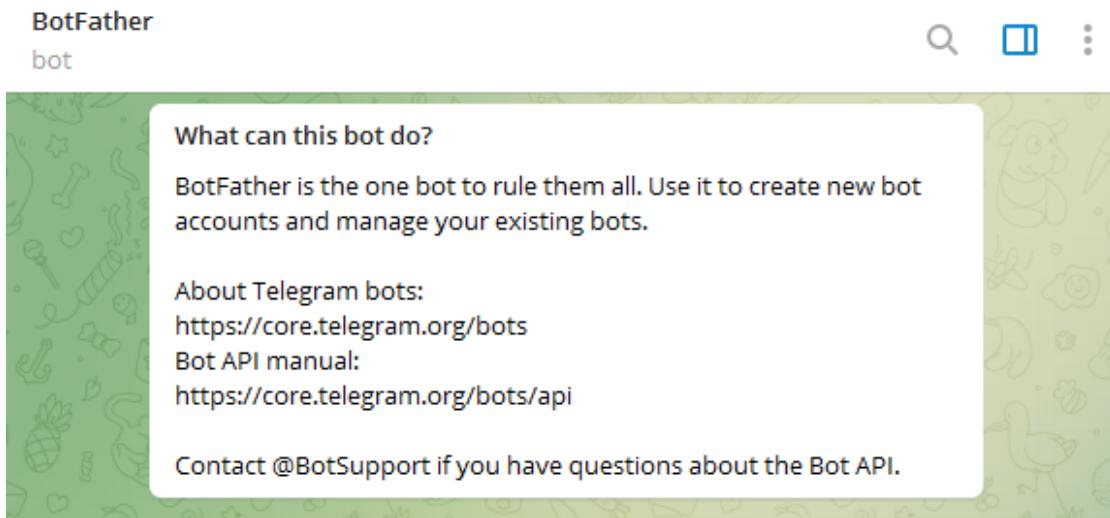
@BotFather



Figura 57: Búsqueda del bot BotFather.

A pesar de que aparecen varias opciones sólo la primera es la que utilizará. Note que posee una tilde azul para diferenciarlo de los demás. Este BotFather no es más que otro bot dedicado exclusivamente a la creación de nuevos bots.

Lo primero que este bot muestra es el siguiente mensaje:



Colocando el comando **/start** mostrará un listado de opciones con las acciones que este bot puede realizar:

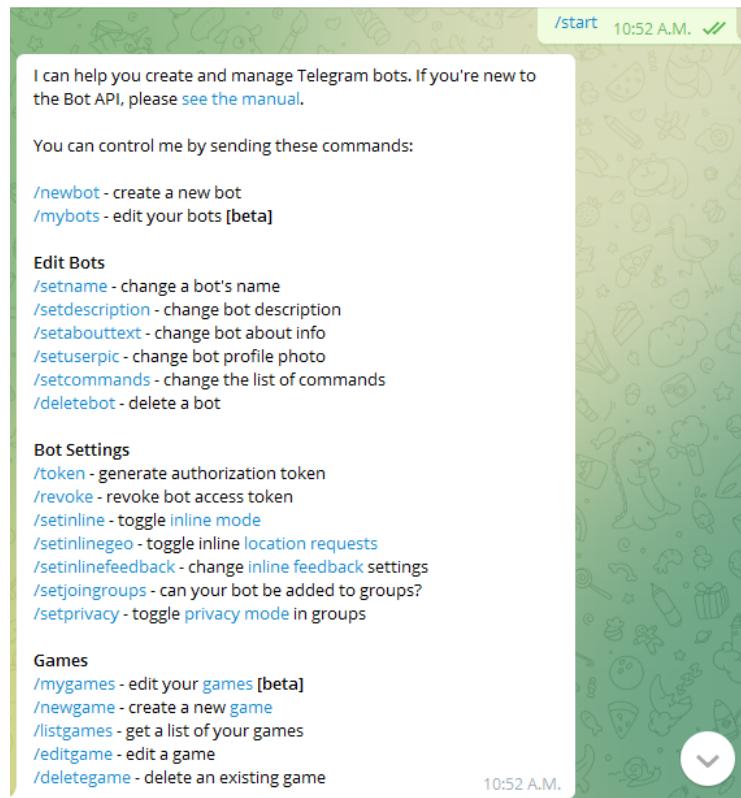


Figura 58: Menú de opciones de BotFather.

Con el comando **/newbot** podrá crear un nuevo bot:



Figura 59: Creación de un bot.

Se debe ingresar un nombre para el bot y un nombre de usuario al mismo.

Una vez realizados estos pasos, ya se ha creado el bot, el cual posee un 'token', que es como una clave de identificación del bot.

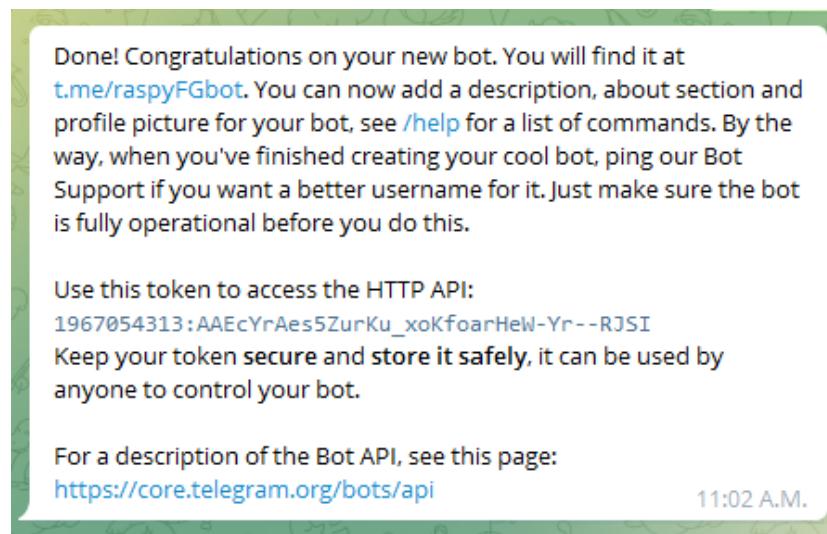


Figura 60: Clave token del bot.

La primera prueba inicial que puede realizar para saber si su bot se ha creado correctamente es enviarle un mensaje desde el navegador. Para ello utilice:

[https://api.telegram.org/bot\[TOKEN\]/sendMessage?chat_id=\[CHAT_ID\]&text\[MENSAJE\]](https://api.telegram.org/bot[TOKEN]/sendMessage?chat_id=[CHAT_ID]&text[MENSAJE])

Por ejemplo:

https://api.telegram.org/bot1967054313:AAEcYrAes5ZurKu_xoKfoarHeW-Yr--RJSI/sendMessage?chat_id=1980654556&text=hola mundo

De haberse enviado correctamente, debería aparecer el siguiente mensaje:

```
{"ok":true,"result": {"message_id":6311,"from": {"id":1967054313,"is_bot":true,"first_name":"raspybot","username":"raspyFGBot"}, "chat": {"id":1980654556,"first_name":"Fer Guerrero", "type": "private"}, "date":1643133576, "text": "hola mundo"}}
```

Y aparecerá en el chat del bot el mensaje enviado:



Para obtener la identificación de su chat puede hacerlo a través del bot `@userinfobot`, el cual al iniciararlo arrojará el chat id:

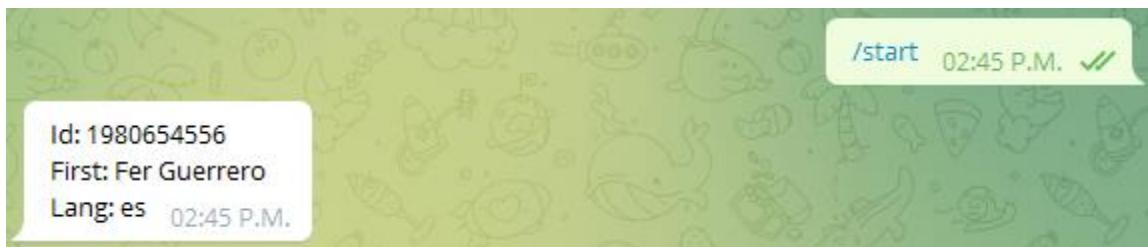


Figura 61: Obtención del ID del chat de usuario.

■ Avisos de inicio

Se ha configurado la placa para que al iniciarse esta o establecer alguna conexión SSH, la misma envíe un mensaje al bot indicando la situación.

Para esto se crean dos scripts. El primero es el script `prueba.sh` encargado de enviar un mensaje cuando la placa se inicia:

```
#!/bin/bash
TOKEN="Coloque aquí el token del bot"
ID="Coloque aquí el ID de su chat"
MENSAJE="Se reinició la placa."
```

```

URL="https://api.telegram.org/bot$TOKEN/sendMessage?chat_id=$ID=
$MENSAJE"
curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE"

```

Genere este script en */home/pi/Desktop*.

Si ejecuta este script se enviará el mensaje “Se reinició la placa” al bot cuyo token se ha colocado. Para que dicho mensaje se envíe automáticamente al inicio de la placa deberá añadir este script a las tareas del Cron de Linux. Estas tareas se ejecutan de forma automática al inicio de la placa, por lo que resultan ideales para el propósito que se busca.

Para ello escriba el siguiente comando en la terminal:

```
sudo crontab -e
```

Y agregue al final del archivo la siguiente línea:

```
@reboot ( sleep 30 ; sh /home/pi/Desktop/prueba.sh ) >/dev/null 2>&1
```

```

# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
@reboot ( sleep 30 ; sh /home/pi/Desktop/prueba.sh ) >/dev/null 2>&1

```

Figura 62: Modificación del archivo crontab.

- @reboot: indica que se ejecute el código que prosigue cada vez que se reinicie el equipo.
- Los paréntesis: Entre ellos se coloca lo que se ejecutará.
- Sleep 30: le indicamos que debe esperar 30 segundos para ejecutar el comando. De esta forma le dará tiempo al sistema operativo a cargar todo lo necesario para ejecutar correctamente el comando.
- sh /home/pi/Desktop/prueba.sh: Es el comando que ejecuta el script llamado prueba.sh.
- >/dev/null 2>&1: aunque esta línea genere algún mensaje en pantalla, no se mostrará.

Al guardar el archivo y reiniciar la placa efectivamente es enviado el mensaje a nuestro bot:



Figura 63: Mensaje de inicio del dispositivo.

El segundo script, llamado *prueba2.sh* envía un mensaje cada vez que alguien se conecta a la placa mediante una conexión SSH. También indica la dirección IP del usuario conectado.

```
#!/bin/bash

TOKEN="1967054313:AAEcYrAes5ZurKu_xoKfoarHeW-Yr--RJSI"
ID="1980654556"
MENSAJE="Alguien se acaba de conectar por SSH con la IP $(echo
$SSH_CLIENT | awk '{print $1}')."
URL="https://api.telegram.org/bot$TOKEN/sendMessage?chat_id=$ID=
$MENSAJE"

curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE" -d
parse_mode=$HTML
```

Genere este script en /home/pi/Desktop.

Cada vez que se inicia una sesión en el terminal, el sistema lee y ejecuta el archivo llamado «.bashrc» (El punto del principio indica que es un archivo oculto). Así que se debe editar este archivo. Ejecute los siguientes comandos:

```
cd /home/pi
sudo nano .bashrc
```

Y agregue al final la línea para ejecutar el script ya visto (*prueba2.sh*):

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi
sh /home/pi/Desktop/prueba2.sh >/dev/null 2>&1
```

Figura 64: Modificación del archivo bashrc.

De esta forma, cada vez que un usuario se conecte a través del programa PuTTY o abra la terminal, se enviará este mensaje:

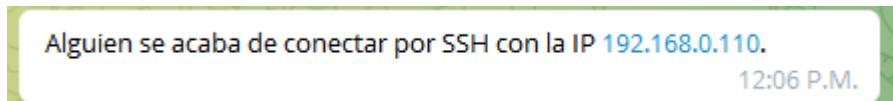


Figura 65: Mensaje enviado en cada conexión SSH.

7.2.1 LIBRERÍAS E INSTRUCCIONES PRINCIPALES

Para lograr la comunicación remota entre el usuario y la Raspberry Pi, se hizo uso de la librería y las funciones obtenidas desde GitHub a través del siguiente link:

<https://github.com/smarnode/telebot>

Debe instalar bibliotecas y herramientas como CMake. En las distribuciones de Linux basadas en Debian, puede hacerlo de la siguiente manera:

```
sudo apt-get install libcurl4-openssl-dev libjson-c-dev cmake  
binutils make
```

Para construir la biblioteca, ejecute los siguientes comandos:

```
cd [su repositorio]  
mkdir -p test && cd test  
cmake ../  
make
```

Las funciones más importantes se describen a continuación.

- **telebot_error_e telebot_create(telebot_handler_t *handle, char *token);**

Esta función debe usarse al inicio para llamar y crear un controlador. Esta llamada debe tener una llamada correspondiente a #telebot_destroy() cuando se complete la operación.

_Parámetro de salida **handle**: es un puntero para crear un controlador de telebot necesario para usar la API; debe destruirse con #telebot_destroy().

_Parámetro de entrada **token**: es la clave que identifica al bot de Telegram creado.

En caso de éxito, se devuelve #TELEBOT_ERROR_NONE, de lo contrario, un valor de error negativo.

- **telebot_error_e telebot_destroy(telebot_handler_t handle);**

Esta función debe ser la última en ser llamada y destruye el controlador creado al comienzo del programa. Es la opuesta a la función #telebot_create().

_Parámetro de entrada **handle**: es el controlador de telebot creado con #telebot_create().

En caso de éxito, se devuelve #TELEBOT_ERROR_NONE, de lo contrario, un valor de error negativo.

- **telebot_error_e telebot_get_updates(telebot_handler_t handle, int offset, int limit, int timeout, telebot_update_type_e allowed_updates[], int allowed_updates_count, telebot_update_t **updates, int *count);**

Esta función se utiliza para obtener las últimas actualizaciones. A través de esta función se obtienen los objetos recibidos. Dentro de estos objetos se poseen los mensajes entrantes.

Esta función se utiliza para obtener las últimas actualizaciones.

_Parámetro de entrada **handle**: es el controlador de telebot creado con #telebot_create().

_Parámetro de entrada **offset**: Identificador de la primera actualización a devolver. El desplazamiento negativo se puede especificar para recuperar actualizaciones a partir del desplazamiento desde el final de la cola de actualizaciones.

_Parámetro de entrada **limit**: Número de actualizaciones a recuperar. Se aceptan valores entre 1-100. El valor predeterminado es 100.

_Parámetro de entrada **timeout**: Tiempo de espera en segundos para sondeos largos. El valor predeterminado es 0, es decir, sondeo breve habitual. En caso de ser positivo, el sondeo corto debe usarse solo con fines de prueba.

_Parámetro de entrada **allow_updates**: Una matriz del tipo de actualizaciones que desea que reciba su bot. Hace referencia a #telebot_update_type_e.

_Parámetro de entrada **allow_updates_count**: Número de tipos de actualización.

_Parámetro de salida **updates**: Una matriz de objetos de actualizaciones recibidas.

_Parámetro de salida **count**: Número de actualizaciones recibidas.

En caso de éxito, se devuelve #TELEBOT_ERROR_NONE, de lo contrario, un valor de error negativo.

- **telebot_error_e telebot_get_me(telebot_handler_t handle, telebot_user_t *me);**

Esta función se utiliza para obtener información sobre el propio bot de Telegram.

_Parámetro de entrada **handle**: El controlador de telebot creado con #telebot_create().

_Parámetro de salida **me**: objeto de usuario de Telegram; debe liberarse con #telebot_put_me después de su uso.

En caso de éxito, se devuelve #TELEBOT_ERROR_NONE y el objeto de usuario se almacena en el parámetro de entrada.

- `telebot_error_e telebot_put_me(telebot_user_t *me);`

Esta función se utiliza para liberar la memoria utilizada para obtener información sobre el bot de Telegram.

_Parámetro de entrada **me**: Puntero al objeto de usuario de Telegram obtenido con `#telebot_get_me`.

En caso de éxito, se devuelve `#TELEBOT_ERROR_NONE`, de lo contrario, un valor de error negativo.

- `telebot_error_e telebot_send_message(telebot_handler_t handle, long long int chat_id, const char *text, const char *parse_mode, bool disable_web_page_preview, bool disable_notification, int reply_to_message_id, const char *reply_markup);`

Función utilizada para enviar mensajes de texto.

_Parámetro de entrada **handle**: El controlador de telebot creado con `#telebot_create()`.

_Parámetro de entrada **chat_id**: Identificador único para el chat de destino o nombre de usuario del canal de destino (en el formato \@channelusername).

_Parámetro de entrada **text**: Texto del mensaje a enviar, 1-4096 caracteres después del análisis de entidades.

_Parámetro de entrada **parse_mode**: Envíe Markdown o HTML, si desea que las aplicaciones de Telegram muestren URL en negrita, cursiva, de ancho fijo o en línea en el mensaje de su bot.

_Parámetro de entrada **disabled_web_page_preview**: Deshabilita las vistas previas de enlaces en el mensaje.

_Parámetro de entrada **disabled_notification**: Envía el mensaje en silencio. Los usuarios recibirán una notificación sin sonido.

_Parámetro de entrada **answer_to_message_id**: Si el mensaje es una respuesta, es el ID del mensaje original.

_Parámetro de entrada **answer_markup**: Opciones de interfaz adicionales. Un objeto para un teclado de respuesta personalizado, instrucciones para ocultar el teclado o forzar una respuesta del usuario.

En caso de éxito se devuelve `#TELEBOT_ERROR_NONE`, de lo contrario, un valor de error negativo.

- `telebot_error_e telebot_send_document(telebot_handler_t handle, long long int chat_id, const char *document, bool is_file, const char *thumb, const char`

```
*caption,const    char    *parse_mode,    bool    disable_notification,    int  
reply_to_message_id,const char *reply_markup);
```

Enviar archivos generales.

_Parámetro de entrada **handle**: El controlador de telebot creado con #telebot_create().

_Parámetro de entrada **chat_id**: Identificador único para el chat de destino o nombre de usuario del canal de destino (en el formato \@channelusername).

_Parámetro de entrada **document**: Archivo de documento a enviar. Es un file_id tipo string para reenviar un archivo que ya está en los servidores de Telegram, o una ruta al archivo.

_Parámetro de entrada **is_file**: Falso si el documento es file_id, verdadero, si el documento es una ruta de archivo.

_Parámetro de entrada **thumb**: Ruta del archivo en miniatura del archivo enviado; se puede ignorar si la generación de miniaturas para el archivo es compatible con el lado del servidor. La miniatura debe estar en formato JPEG y tener un tamaño inferior a 200 kB. El ancho y la altura de una miniatura no deben exceder los 320. Opcional. Establecer en NULL para ignorar.

_Parámetro de entrada **caption**: Título del documento. (también se puede utilizar al reenviar documentos).

_Parámetro de entrada **parse_mode**: Envíe Markdown o HTML, si desea que las aplicaciones de Telegram muestren URL en negrita, cursiva, de ancho fijo o en línea en el mensaje de su bot.

_Parámetro de entrada **disabled_notification**: Envía el mensaje en silencio. Los usuarios recibirán una notificación sin sonido.

_Parámetro de entrada **answer_to_message_id**: Si el mensaje es una respuesta, es el ID del mensaje original.

_Parámetro de entrada **answer_markup**: Opciones de interfaz adicionales. Un objeto para un teclado de respuesta personalizado, instrucciones para ocultar el teclado o forzar una respuesta del usuario.

En caso de éxito, se devuelve #TELEBOT_ERROR_NONE, de lo contrario, un valor de error negativo.

7.2.2 PROGRAMA PARA COMUNICACIÓN REMOTA DESDE TELEGRAM

Se desarrolló un programa capaz de mostrarle al usuario un menú de opciones e interactuar con el dispositivo a distancia. El mismo le da la capacidad al usuario de descargar los archivos generados con las muestras tomadas, ver un listado completo de dichos archivos, ver y modificar los parámetros de configuración y testear la temperatura de la placa.

A continuación, se detallan las variables y funciones implementadas.

- **Librerías utilizadas:**

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#include <telebot.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
```

- **Defines y variables globales:**

```
/*/////////////////DEFINES////////////////*/  
  
#define SIZE_OF_ARRAY(array) (sizeof(array) / sizeof(array[0]))  
#define FIFO_PATH "/tmp/MI_FIFO"  
  
/*/////////////////Definición de variables globales////////////////*/  
  
int flag1=3;           //Variable bandera 1  
int flag2=3;           //Variable bandera 2  
int RetCode=0;          //Variable de retorno para control del programa  
int cantidad_archivos; //Entero donde se obtendrá la cantidad de archivos  
                        //almacenables  
int lastfile;          //Entero con el número del último archivo creado  
int firstfile;          //Entero con el número del archivo creado más antiguo  
int number_file2;       //Entero que llevará el conteo de archivos  
int lista;              //Variable bandera para el listado de archivos  
int nread;              //Entero con la cantidad de caracteres leídos de la FIFO  
int fifo_t;             //Descriptor de archivo para la apertura de la FIFO  
int err;                //Variable de retorno en la creación de la FIFO  
int pid_m;              //Entero con el PID del proceso de medición  
char yourpid[20];        //Arreglo donde obtendremos el PID del proceso de medición  
char auxiliar_cantidad[50]={""}; //Arreglo auxiliar donde obtendremos la  
                                //cantidad de archivos almacenables  
  
telebot_handler_t handle; //Este es un objeto opaco para representar a un  
                          //manipulador de telebot.  
telebot_user_t me;        //Este objeto representa un usuario o bot de Telegram.  
telebot_error_e ret;      //Variable de retorno para control del programa. En caso de  
                          //éxito, se devuelve #TELEBOT_ERROR_NONE;  
                          //de lo contrario, un valor de error negativo.  
telebot_message_t message; //Este objeto representa un mensaje.
```

```

telebot_update_t *updates; //Este objeto representa una actualización entrante.
telebot_update_type_e update_types[] = {TELEBOT_UPDATE_TYPE_MESSAGE};
//Representa el tipo de actualización.

```

- **Función atoi:**

```

/*/////////////////////////////Función "atoifunction"////////////////////////////*/
int atoifunction(char *character) //Función que permite implementar la función
                                    atoi dentro de una estructura condicional
{
    return atoi(character); //Función atoi que permite obtener el valor numérico
                            del arreglo de caracteres pasado como argumento
}

```

- **Función obtener cantidad:**

Ya sea al listar archivos, bajar el último archivo generado o los últimos 10, se puede requerir determinar cuál es el mayor archivo numérico y si este ha sido generado o no. Esta función lee el parámetro “cantidad de archivos almacenables”, debido a que esta es la cantidad máxima de archivos que pueden generarse.

Diagrama de flujo:

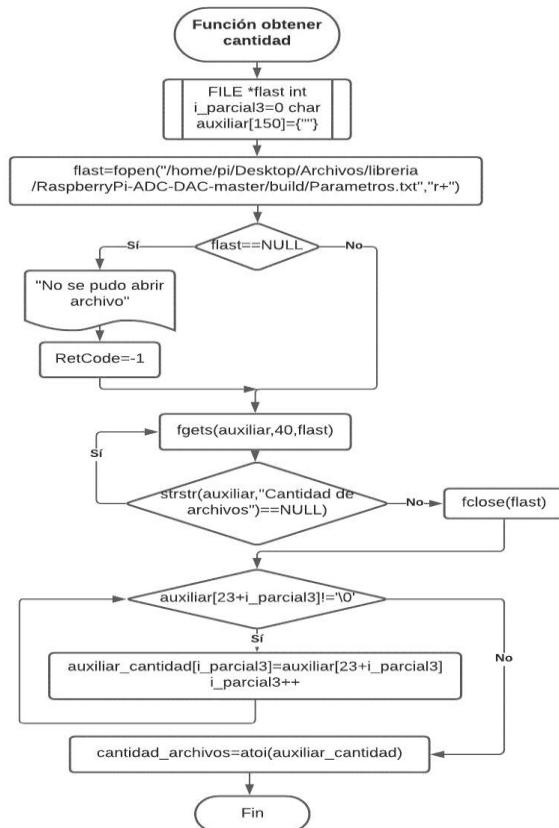


Figura 66: Diagrama de flujo de la función “obtener_cantidad”.

Código fuente:

```
///////////Función "obtener_cantidad"///////////  
1 void obtener_cantidad() //Función que obtiene la cantidad de archivos  
    almacenables  
2 {  
3     FILE *flast; //Descriptor del archivo Parametros.txt  
4     int i_parcial3=0; //Entero auxiliar  
5     char auxiliar[150]={""]; //Arreglo auxiliar  
6     flast=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-  
        DAC/build/Parametros.txt","r+"); //Apertura del archivo Parametros.txt  
7     if(flast==NULL) //Si flast=NULL, no pudo abrirse correctamente  
        el archivo Parametros.txt  
8     {  
9         printf("No se pudo abrir archivo\n");  
10        RetCode=-1; //Si no pudo abrirse el archivo RetCode toma el valor -1  
11    }  
12    do //Mientras la línea leída sea distinta de "Cantidad de archivos"  
13    { //obtendrá línea por línea  
14        fgets(auxiliar,40,flast);  
15    } while(strstr(auxiliar,"Cantidad de archivos")==NULL);  
16    fclose(flast); //Cerramos el descriptor de archivo  
17    while(auxiliar[23+i_parcial3]!='\0') //A partir de la posición 23 hasta el  
        final del arreglo obtenemos la cantidad de archivos y lo guardamos en  
        auxiliar_cantidad  
18    {  
19        auxiliar_cantidad[i_parcial3]=auxiliar[23+i_parcial3]  
20        i_parcial3++;  
21    }  
22    cantidad_archivos=atoi(auxiliar_cantidad); //Obtenemos numéricamente la  
        cantidad de archivos  
23 }
```

Consideraciones: en la obtención del parámetro en la línea 19 se ha tenido en cuenta que el número de archivos almacenables comienza en la posición 23 de la sexta línea del archivo de configuración. Si modifica esta línea agregando o sacando caracteres anteriores al parámetro, no se obtendrá el número correcto.

▪ Función enviar archivo:

Esta función es utilizada cada vez que se desea enviar un archivo al bot. Recibe el nombre y ruta del archivo a enviar, elimina los caracteres innecesarios e implementa la función necesaria para envío de documentos por Telegram.

Diagrama de flujo:

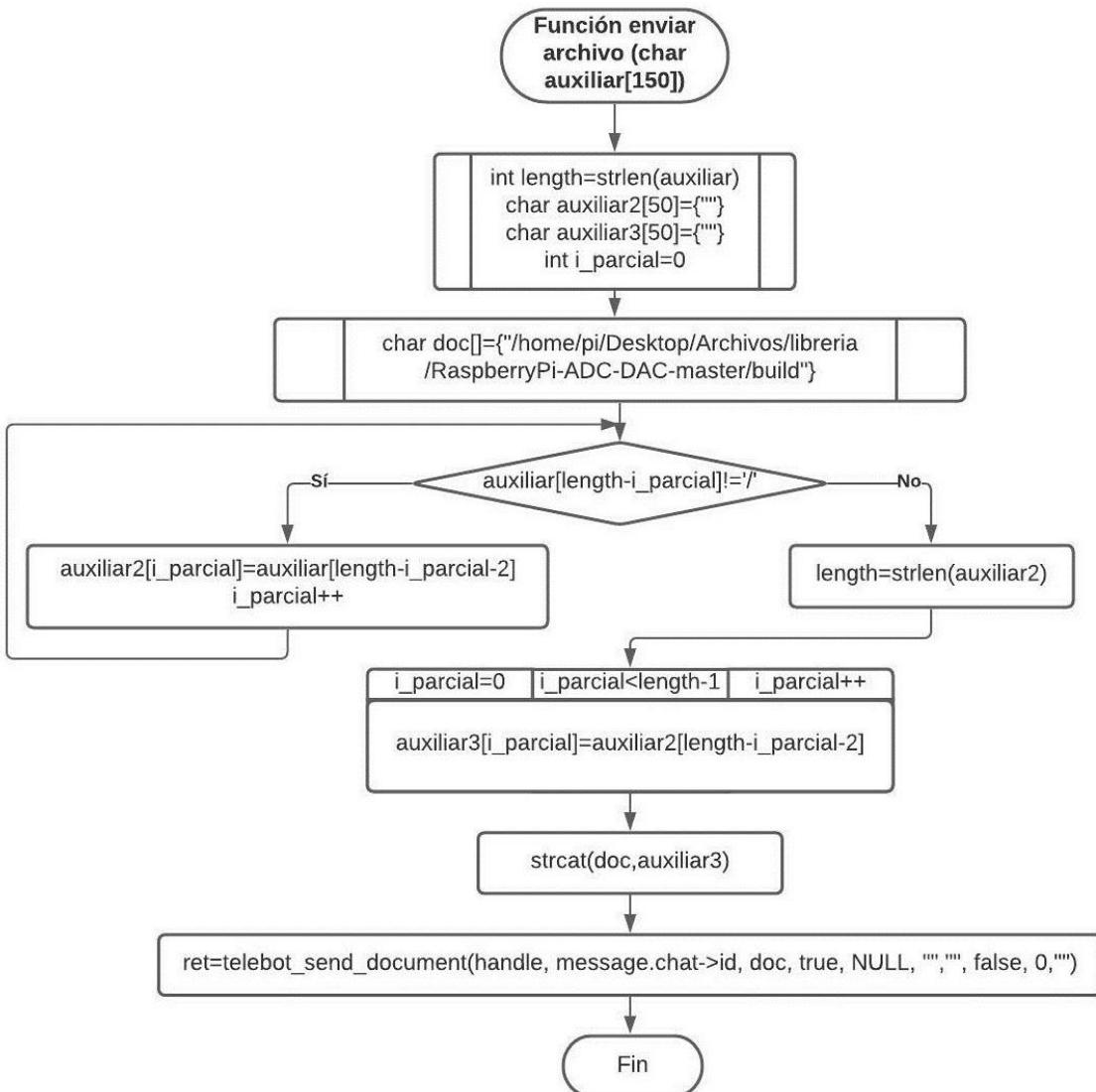


Figura 67: Diagrama de flujo de la función “enviar_archivo”.

Código fuente:

```

/*
 * Función "enviar_archivo"
 */
1 void enviar_archivo(char auxiliar[300])      //Función que envía el archivo cuyo
                                                //nombre se pasa en su argumento
2 {
3     int length=strlen(auxiliar);             //Longitud del arreglo auxiliar de entrada
4     char auxiliar2[50]={""};                 //2do arreglo auxiliar
5     char auxiliar3[50]={""};                 //3er arreglo auxiliar
6     char doc[]={"/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build"};
                                                //Arreglo con la ruta donde se encuentran los archivos a enviar
  
```

```

7   char doc2[400];           //Arreglo auxiliar a enviar
8   int i_parcial=0;          //Entero auxiliar

9   strcpy(doc2,doc);
10  while(auxiliar[length-i_parcial]!='/') //Recorremos el arreglo auxiliar desde
11    el final hasta encontrar el carácter
12    '/'
13
14  {
15    auxiliar2[i_parcial]=auxiliar[length-i_parcial-2];
16    //En auxiliar2 guardamos el nombre invertido del archivo a enviar
17    i_parcial++;
18  }
19
20  length=strlen(auxiliar2);           //Longitud del nombre del archivo
21  for(i_parcial=0;i_parcial<length-1;i_parcial++)
22    //Invertimos el arreglo auxiliar2 y obtenemos el nombre correcto en auxiliar3
23  {
24    auxiliar3[i_parcial]=auxiliar2[length-i_parcial-2];
25  }
26
27  strcat(doc2,auxiliar3);           //Concatenamos el nombre del archivo con la ruta
28  donde se encuentra
29
30  ret=telebot_send_document(handle, message.chat->id, doc2, true, NULL, "", "", false, 0, ""); //Enviamos el documento especificado en el tercer argumento
31
32

```

Consideraciones: la razón por la que antes de enviar el archivo se separa solamente el nombre del archivo y luego se concatena a la ruta donde se encuentra dicho archivo es la siguiente. Por lo general al buscar un archivo con el comando *find*, el resultado de esta búsqueda es almacenado en el archivo “resultado.txt”. Esta búsqueda arroja el nombre del archivo en cuestión, su ruta y, además, la fecha de creación y otros datos cuya longitud es impredecible. Toda esta línea obtenida del archivo es pasada al argumento de la función enviar archivo, por lo que no puede simplemente utilizar el argumento para enviar el archivo especificado, si no que primero debe hacerse una limpieza y posteriormente enviar el archivo.

- Función último:

La primera de las opciones que se le da al usuario dentro del menú de bajar archivos es bajar el último archivo.

Esta función encuentra el último archivo creado leyendo el archivo lastfile.txt que, recordemos, contiene el número del archivo siguiente al último creado. Por lo tanto, resta uno a este número, realiza una búsqueda del archivo que comienza con dicho número en la ruta especificada y guarda el resultado en el archivo resultado.txt. Por último, lee este archivo y llama a la función enviar_archivo pasándole como argumento el resultado de la búsqueda.

Diagrama de flujo:

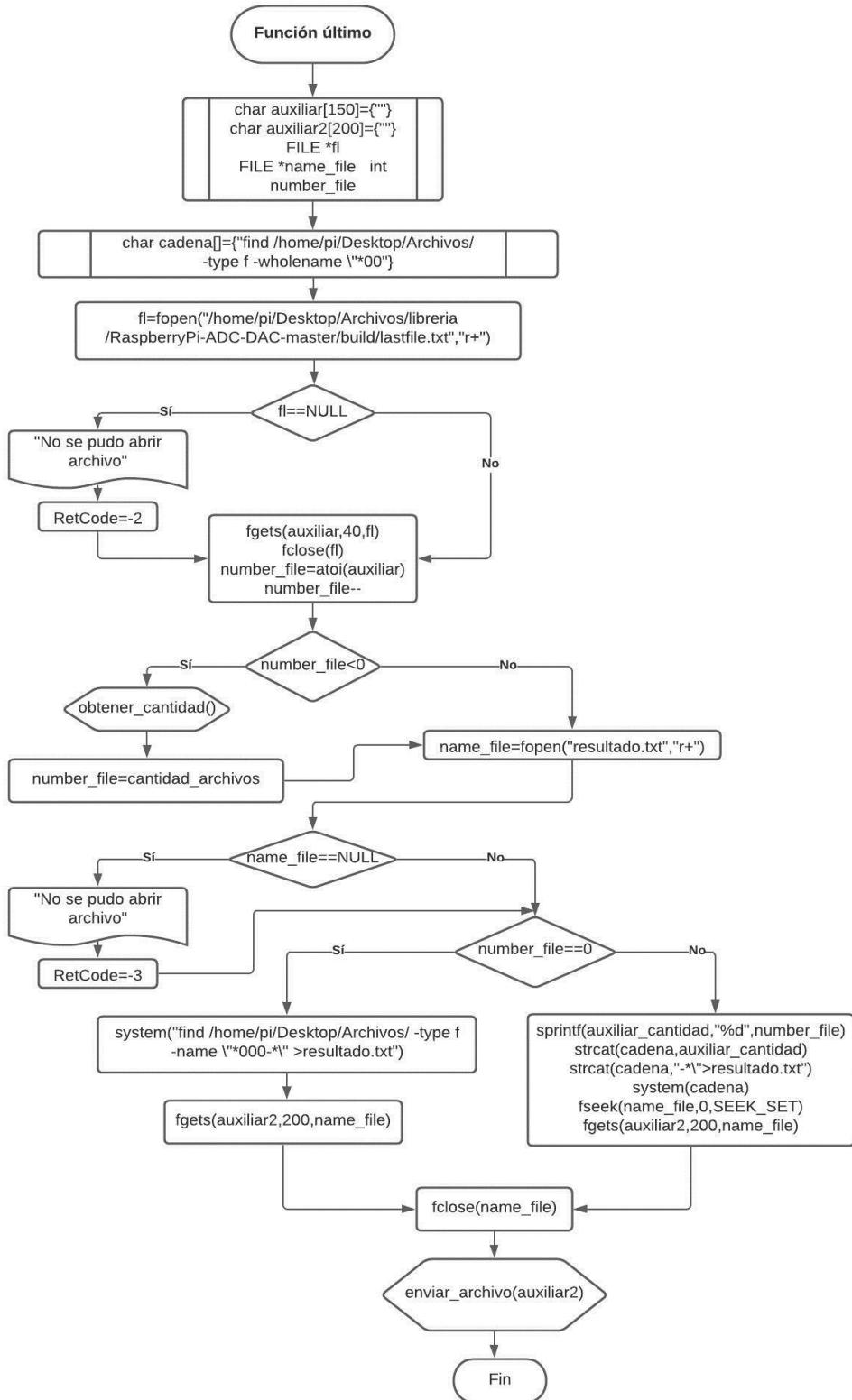


Figura 68: Diagrama de flujo de la función “ultimo”.

Código fuente:

```
/*/////////////////////////////Función "ultimo"////////////////////////////*/\n\n1 void ultimo() //Función que envía el último archivo creado\n2 {\n3     char auxiliar[150]={"\"}; //Arreglo auxiliar\n4     char auxiliar2[200]={"\"}; //2do arreglo auxiliar\n5     char cadena[]={\"find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-\n        DAC/build/ -type f -wholename \"*00\"\"}; //Arreglo con el\n        comando y la ruta donde buscaremos el archivo\n6     FILE *fl; //Descriptor para el archivo lastfile.txt\n7     FILE *name_file; //Descriptor para el archivo resultado.txt\n8     int number_file; //Entero donde guardaremos el número del último archivo\n9     fl=fopen(\"/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC-\n        /build/lastfile.txt\",\"r+\""); //Apertura del archivo lastfile.txt\n10    if(fl==NULL) //Si fl=NULL, no pudo abrirse el archivo lastfile.txt\n11    {\n12        printf(\"No se pudo abrir archivo\\n\");\n13        RetCode=-2; //En caso de no poder abrir el archivo, RetCode toma el valor -2\n14    }\n15    fgets(auxiliar,40,fl); //Obtención del número del archivo siguiente al\n        último creado\n16    fclose(fl); //Cierre del descriptor de archivo\n17    number_file=atoi(auxiliar); //Obtenemos numéricamente el número del\n        archivo siguiente al último\n18    number_file--; //Restamos en 1 para obtener el número del último archivo\n19    if(number_file<0) //Si number_file es negativo, significa que el\n        número obtenido de lastfile es igual a 0, por lo\n        tanto, el archivo anterior es el mayor en número\n20    {\n21        obtener_cantidad(); //Obtención de la cantidad de archivos almacenables\n22        number_file=cantidad_archivos; //number_file toma el valor de\n            cantidad_archivos\n23    }\n24    name_file=fopen(\"/home/pi/Desktop/Archivos/telegrambot/telebot-\n        master/test/test/resultado.txt\",\"r+\""); //Apertura del archivo resultado.txt\n25    if(name_file==NULL) //Si name_file=NULL, no pudo abrirse el\n        archivo resultado.txt\n26    {\n27        printf(\"No se pudo abrir archivo\\n\");\n28        RetCode=-3; //En caso de no poder abrir el archivo, RetCode toma el valor -3\n29    }\n30    if(number_file==0) //Si number_file=0 se envía el archivo '000'\n31    {\n32        system(\"find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/ -type f\n            -name \"*000-*\" >/home/pi/Desktop/Archivos/telegrambot/telebot-\n            master/test/test/resultado.txt\"); //Ejecución del comando find que buscará\n            el archivo
```

```

33 fgets(auxiliar2,200,name_file);           //cuyo nombre empiece por '000' en la
34 }                                         ruta especificada y guardará el nombre
35 else                                     completo en el archivo resultado.txt
36 {
37     sprintf(auxiliar_cantidad,"%d",number_file); //Conversión de number_file a
38     strcat(cadena,auxiliar_cantidad);           //Concatenación del comando find, ruta
39     strcat(cadena,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
39         master/test/test/resultado.txt"); //Concatenación de la cadena anterior
39         con el nombre del archivo donde se guarda el resultado de la búsqueda
40     system(cadena);                         //Aplicación del comando a través de la función system
41     fseek(name_file,0,SEEK_SET);            //Reubicamos el puntero del archivo al inicio
41         del mismo
42     fgets(auxiliar2,200,name_file); //Leemos la ruta y el nombre del último archivo
43 }
44 fclose(name_file);                      //Cierre del archivo resultado.txt
45 enviar_archivo(auxiliar2);             //Paso de la ruta y nombre del último archivo como
45         argumento a la función que enviará el archivo
46 }

```

Consideraciones:

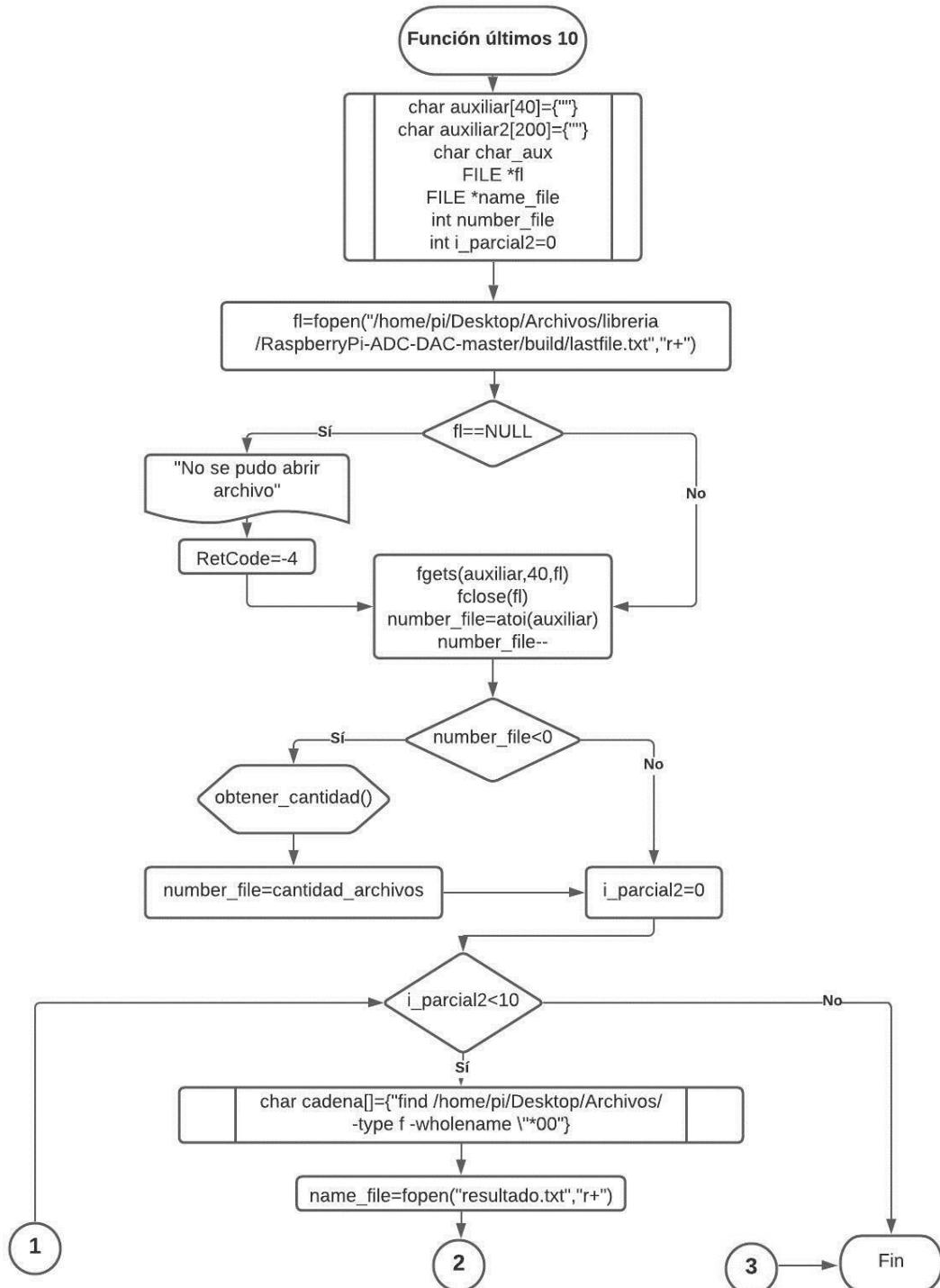
- *El caso en que el archivo a enviar sea el número 0 es considerado de forma independiente. Esto se debe a que en la creación de los archivos siempre se anteponen dos ceros al número de archivo en su nombre y al concatenar estos dos ceros con el número 0 del número de archivo, el resultado no era '000', si no '00'. Por esta razón la búsqueda fallaba y se implementó la alternativa de buscar en forma directa un archivo cuyo nombre comenzara con '000'.*
- *Al llamar a la función system, la cual permite aplicar el comando find para la búsqueda de archivos, el resultado de dicha búsqueda debe ser almacenado en un archivo para su posterior lectura. Esto es debido a que la función system permite aplicar el comando especificado, pero no retorna el resultado de ese comando, si no 0 o -1 dependiendo si la aplicación del comando fue exitosa o no.*
- *Para minimizar la probabilidad de encontrar dos archivos cuyo nombre comience con el mismo número, se decidió anteponer dos ceros al número de archivo y también realizar la búsqueda considerando el carácter '-' luego del número.*

▪ Función últimos10:

La segunda opción de descarga de archivos que se le presenta al usuario es descargar los últimos 10 archivos generados. El mecanismo es el mismo que el empleado para la función anterior, con la diferencia de que en este caso se envían los archivos desde el más nuevo hasta 10 archivos anteriores dentro de un bucle for. De la misma forma, se tiene en cuenta el caso

especial del archivo número cero para modificar el contador del archivo al mayor numéricamente, en caso de que exista, caso contrario se dejan de enviar archivos.

Diagrama de flujo:



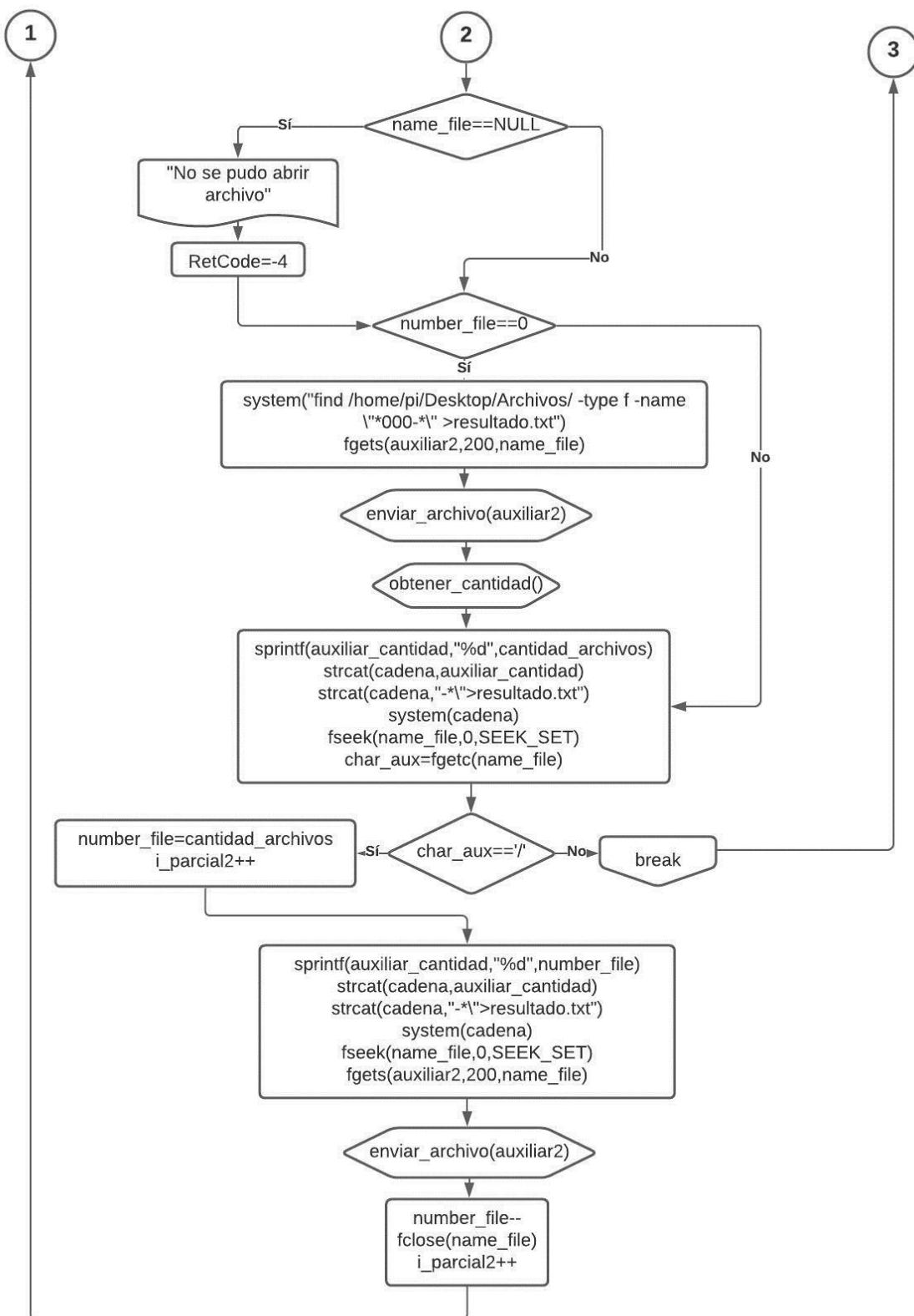


Figura 69: Diagrama de flujo de la función “ultimos10”.

Código fuente:

```
/*/////////////////////////////Función "ultimos10"////////////////////////////*/
1 void ultimos10() //Función que envía los 10 archivos más nuevos
2 {
3     FILE *fl; //Descriptor para el archivo lastfile.txt
4     FILE *name_file; //Descriptor para el archivo resultado.txt
5     int i_parcial2=0; //Entero auxiliar
6     int number_file; //Entero con el número del archivo a enviar
7     char auxiliar[40]={" "}; //Arreglo auxiliar
8     char auxiliar2[200]={" "}; //2do arreglo auxiliar
9     char char_aux; //Carácter auxiliar
10    fl=fopen("/home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-
11        DAC/build/lastfile.txt","r+"); //Apertura del archivo lastfile.txt en modo
12        lectura/escritura
13    if(fl==NULL) //Si fl=NULL no pudo abrirse el archivo lastfile.txt
14    {
15        printf("No se pudo abrir archivo\n");
16        RetCode=-4; //En caso de no poder abrir el archivo, RetCode toma el valor -4
17    }
18    fgets(auxiliar,40,fl); //Obtención del número del archivo siguiente al último creado
19    fclose(fl); //Cierre del descriptor del archivo lastfile.txt
20    number_file=atoi(auxiliar); //Obtenemos numéricamente el número del archivo
21        siguiente al último
22    number_file--; //Restamos en 1 para obtener el número del último archivo
23    if(number_file<0) //Si number_file es negativo, significa que el número
24        obtenido de lastfile es igual a 0, por lo tanto, el
25        archivo anterior es el mayor en número
26    { obtener_cantidad(); //Obtención de la cantidad de archivos almacenables
27        number_file=cantidad_archivos; //number_file toma el valor de cantidad_archivos
28    }
29    for(i_parcial2=0;i_parcial2<10;i_parcial2++) //Bucle for que busca y envía los 10
30        archivos
31    {
32        char cadena[]={ "find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/
33            -type f -wholename \"*00\""; //Areglo con el comando y la ruta donde
34            buscaremos el archivo
35        name_file=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
36            master/test/test/resultado.txt","r+"); //Apertura del archivo resultado.txt
37        if(name_file==NULL) //Si name_file=NULL no pudo abrirse el archivo resultado.txt
38        {
39            printf("No se pudo abrir archivo\n");
40            RetCode=-4; //En caso de no poder abrir el archivo, RetCode toma el valor -4
41        }
42        if(number_file==0) //Si number_file=0 debemos continuar enviando desde el
43            mayor archivo numérico, en caso de que exista
44        {
45            system("find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/ -type f
46            -name \"*000-*\" >/home/pi/Desktop/Archivos/telegrambot/telebot-
47            master/test/test/resultado.txt"); //Búsqueda del archivo '000' y su nombre y ruta se
48            guardan en el archivo resultado.txt
49        fgets(auxiliar2,200,name_file); //Lectura del nombre y ruta del archivo '000'
```

```

36 enviar_archivo(auxiliar2);           //Dicho nombre y ruta es pasado como argumento a la
                                         función que enviará el archivo
37 obtener_cantidad();                //Obtención de la cantidad máxima de archivos almacenables
38 sprintf(auxiliar_cantidad,"%d",cantidad_archivos);
                                         //Conversión de la cantidad máxima de archivos a un arreglo de caracteres
39 strcat(cadena,auxiliar_cantidad);    //Concatenación del comando find, ruta de
                                         búsqueda y número de archivo
40 strcat(cadena,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
                         master/test/test/resultado.txt"); //Concatenación de la cadena resultante
                                         anterior con el nombre del archivo donde se guardará el resultado
41 system(cadena);                   //Aplicación del comando
42 fseek(name_file,0,SEEK_SET);       //Reubicación del puntero del archivo al inicio
43 char_aux=fgetc(name_file);        //Obtención del primer carácter
44 if(char_aux=='/')               //Si el primer carácter es '/' significa que existe un
                                         archivo con el nombre buscado
45 { number_file=cantidad_archivos; //number_file toma el valor del mayor archivo
46 i_parcial2++; }
47 else
48 { break; } //Si el primer carácter es distinto de '/', no se encontró ningún archivo
                                         con el nombre buscado, por lo tanto, estamos en el caso de que existen
                                         menos de 10 archivos almacenados y ya se enviaron todos
49 }
50 sprintf(auxiliar_cantidad,"%d",number_file); //Conversión de number_file a un
                                         arreglo de caracteres
51 strcat(cadena,auxiliar_cantidad); //Concatenación del comando find, ruta de
                                         búsqueda y número de archivo
52 strcat(cadena,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
                         master/test/test/resultado.txt"); //Concatenación de la cadena
                                         resultante anterior con el nombre del archivo donde se guardará el resultado
53 system(cadena);                   //Aplicación del comando
54 fseek(name_file,0,SEEK_SET);       //Reubicación del puntero del archivo al inicio
55 fgets(auxiliar2,200,name_file);   //Lectura del nombre y ruta del archivo
56 enviar_archivo(auxiliar2);        //Dicho nombre y ruta es pasado como argumento a la
                                         función que enviará el archivo
57 number_file--;
58 fclose(name_file);              //Disminución en 1 del número de archivo a enviar
                                         //Cierre del descriptor de archivo
59 }
60 }

```

Consideraciones:

- En la línea 25 el arreglo ‘cadena’ se declara dentro del bucle for, ya que, en caso contrario, en cada iteración pueden quedar caracteres de la búsqueda anterior y no permite la aplicación correcta del comando find. De esta forma, se reinicia el arreglo en cada iteración.
- De la línea 37 a la 41 se busca el mayor archivo numérico. Si este existe, el primer carácter leído del archivo “resultado.txt” será ‘/’, indicando que ya se han generado todos los archivos desde el número 0 a la cantidad de archivos almacenables y por lo tanto el contador de archivos debe continuar por el siguiente al mayor. Caso contrario, el archivo

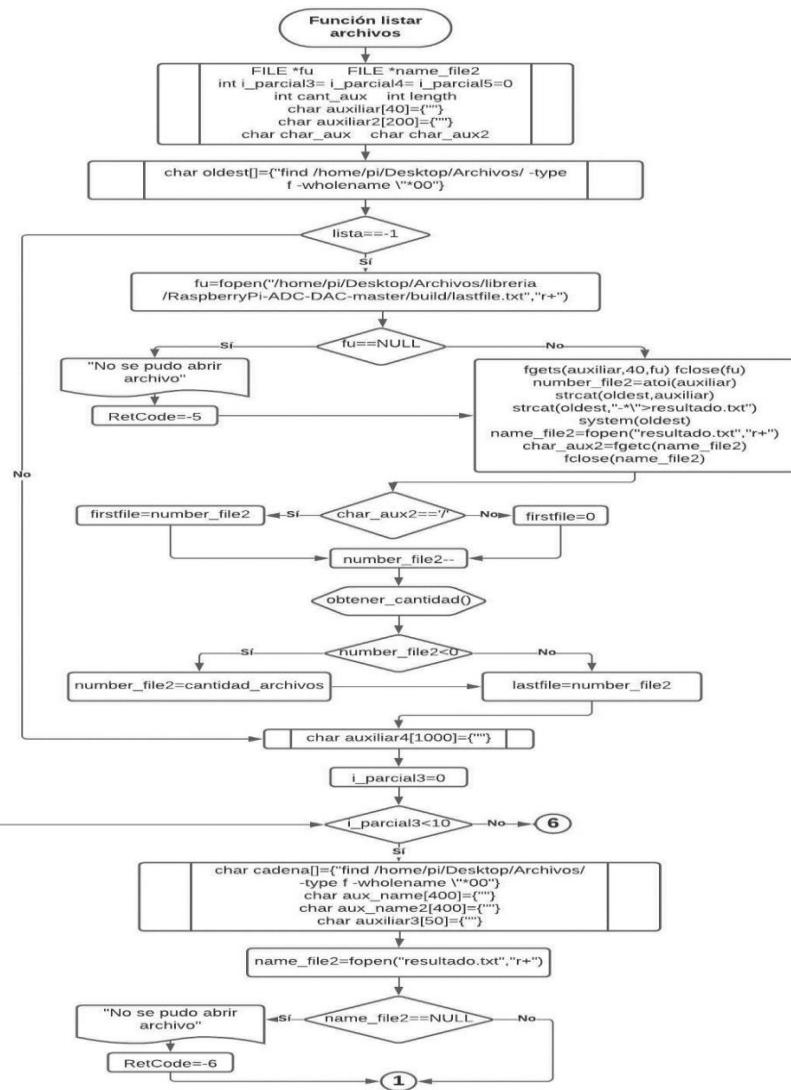
"resultado.txt estará vacío, indicando el caso en que hay menos de 10 archivos almacenados y por lo tanto ya se enviaron todos y se debe salir del bucle.

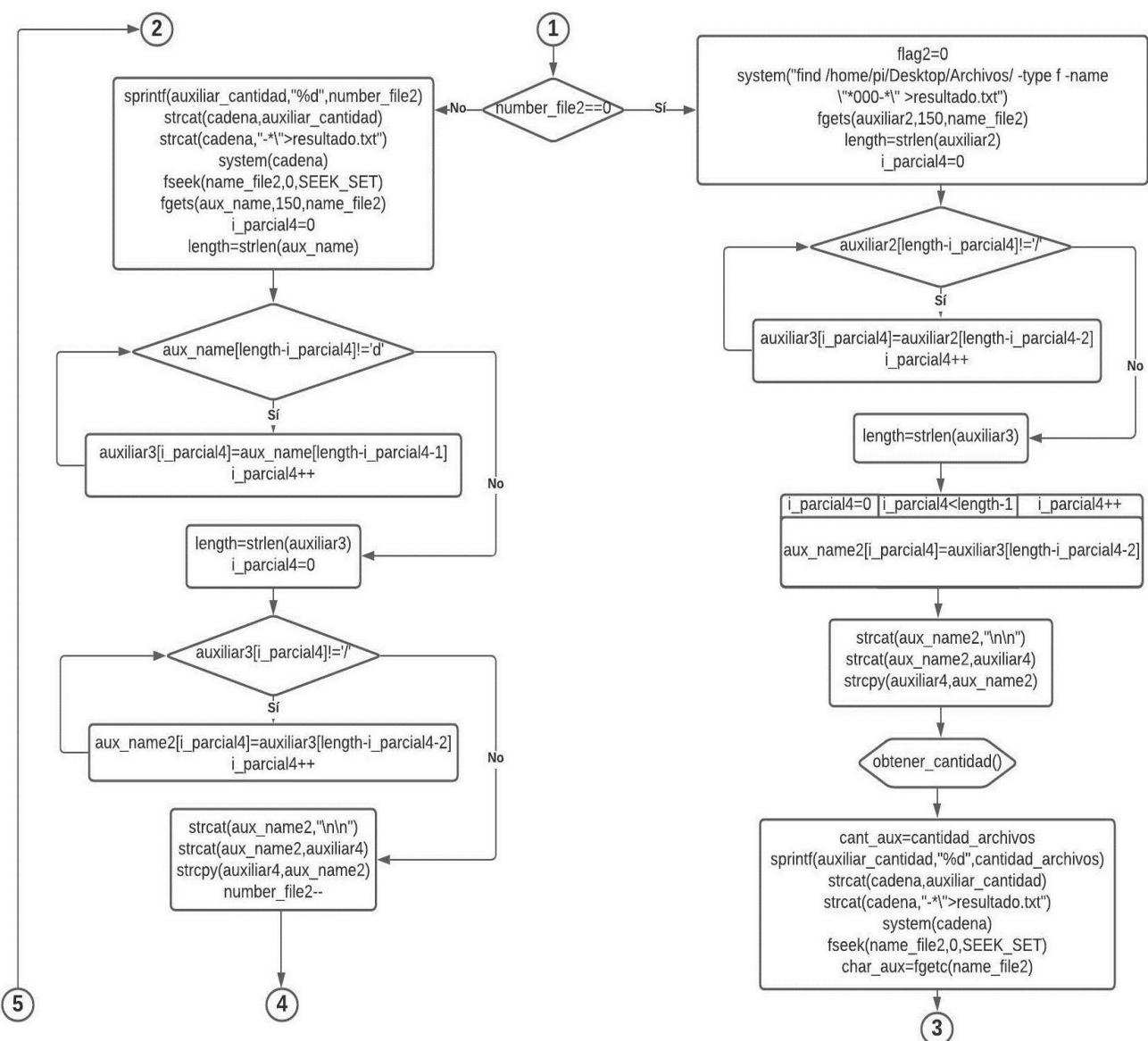
- Debe tener en cuenta que el mecanismo de búsqueda planteado se basa en partir del último archivo generado e ir buscando numéricamente el archivo anterior. Si usted quita archivos intermedios o cambia la cantidad de archivos almacenables a un número menor sin retirar los de mayor numeración, dichos archivos no serán enviados.

▪ Función listar archivos:

La tercera opción que se le da al usuario para la transferencia de archivos es listar los archivos generados. La primera vez que se llama a esta función se listarán los 10 últimos archivos y adicionalmente el usuario podrá listar los 10 archivos numéricos posteriores o los 10 anteriores. De esta forma podrá ver todos los archivos existentes en grupos de 10.

Diagrama de flujo:





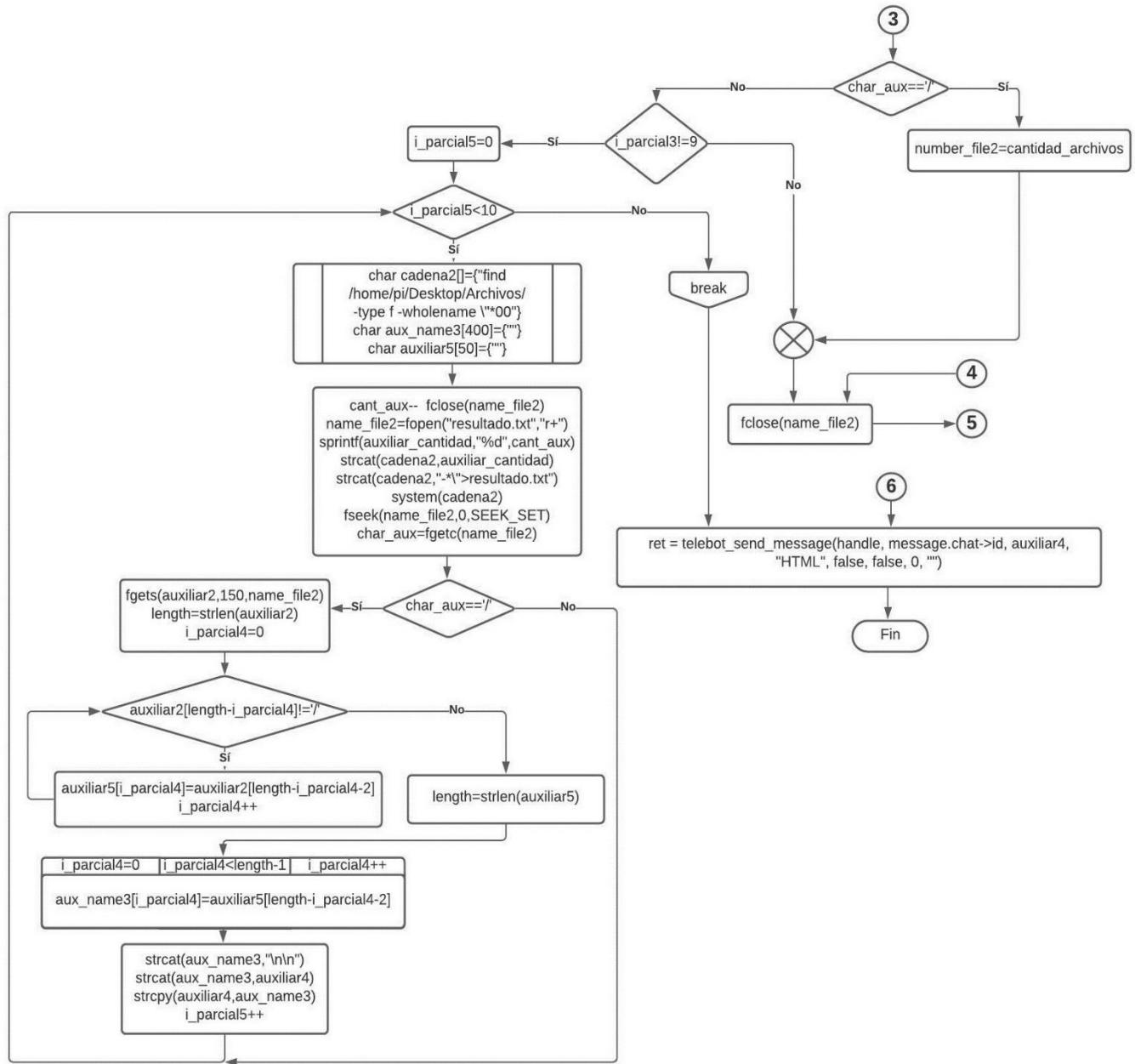


Figura 70: Diagrama de flujo de la función “listar_archivos”.

Código fuente:

```

/*
Función "listar_archivos"
1 void listar_archivos()           //Función que muestra el listado de archivos y permite
                                    //mostrar los 10 archivos siguientes o los 10 anteriores
2 {
3     FILE *fu;                   //Descriptor para el archivo lastfile.txt
4     FILE *name_file2;           //Descriptor para el archivo resultado.txt
5     int i_parcial3=0;           //Entero auxiliar

```

```

5  int i_parcial4=0;                                //2do entero auxiliar
6  int i_parcial5=0;                                //3er entero auxiliar
7  int cant_aux;                                    //Entero auxiliar para cantidad
8  int length;                                     //Entero auxiliar para longitud de arreglos
9  char auxiliar[40]={""};                          //Arreglo auxiliar
10 char auxiliar2[200]={""};                         //2do arreglo auxiliar
11 char char_aux;                                  //Carácter auxiliar
12 char char_aux2;                                 //2do carácter auxiliar
13 char oldest[]={ "find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/ -"
   type f -wholename \"*00\"};           //Arreglo con el comando y la ruta donde buscaremos el
   archivo
14 if(lista== -1)                               //Si lista=-1, estamos en el primer caso donde hemos
   llamado a la función listar_archivos
15 {
16  fu=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC
   /build/lastfile.txt","r+");    //Apertura del archivo lastfile.txt
17  if(fu==NULL)                                //Si fu=NULL, no pudo abrirse el archivo lastfile.txt
18  {
19    printf("No se pudo abrir archivo\n");
20    RetCode=-5;      //Si no pudo abrirse el archivo lastfile.txt, RetCode toma el valor -5
21  }
22  fgets(auxiliar,40,fu);                     //Lectura del número del archivo siguiente al último
23  fclose(fu);                                //Cierre del archivo lastfile.txt
24  number_file2=atoi(auxiliar);                //Obtención numérica del siguiente archivo al
   último creado
25  strcat(oldest,auxiliar);                   //Concatenación del comando find, la ruta de búsqueda
   y el número del archivo
26  strcat(oldest,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
   master/test/test/resultado.txt");        //Concatenación del arreglo
   resultante anterior y el nombre del archivo resultado.txt
27  system(oldest);                            //Aplicación del comando
28  name_file2=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
   master/test/test/resultado.txt","r+");    //Apertura del archivo resultado.txt
29  char_aux2=fgetc(name_file2);              //Obtención del primer carácter del archivo
30  fclose(name_file2);                      //Cierre del archivo resultado.txt
31  if(char_aux2=='/')          //Si el primer carácter es '/', existe el archivo que le sigue
   numéricamente al último generado
32  {
33    firstfile=number_file2;                  //El entero firstfile toma el valor numérico del
   archivo más antiguo
34  }
35 else
36 {firstfile=0;                                //Si el archivo siguiente al más nuevo no existe,
   el más antiguo es el 0
37 }
38 number_file2--;                            //Disminución en 1 de number_file2 para obtener el número del
   último archivo creado
39 obtener_cantidad();                        //Llamada a la función que obtiene la cantidad de
   archivos almacenables
40 if(number_file2<0)                         //Si number_file2 es negativo, significa que el archivo
   más nuevo es el 0
41 {

```

```

42     number_file2=cantidad_archivos;           //Por lo anterior, number_file2 toma el valor
                                                del mayor archivo numérico
43 }
44 lastfile=number_file2;           //lastfile toma el valor numérico del archivo más nuevo
45 }
46 char auxiliar4[1000]={""};          //4to arreglo auxiliar
47 for(i_parcial3=0;i_parcial3<10;i_parcial3++) //Bucle for que enviará los nombres
                                                de los 10 archivos
48 {
49     char cadena[]={find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/
      -type f -wholename \'*00'}; //Arreglo con el comando y la ruta donde buscaremos el
      archivo
50     char aux_name[400]={""};           //Arreglo auxiliar para el nombre del archivo
51     char aux_name2[400]={""};          //2do arreglo auxiliar para el nombre del archivo
52     char auxiliar3[50]={""};           //3er arreglo auxiliar
53     name_file2=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
      master/test/test/resultado.txt","r+"); //Apertura del archivo resultado.txt
54     if(name_file2==NULL)   //Si name_file2=NULL, no pudo abrirse el archivo resultado.txt
55     {
56         printf("No se pudo abrir archivo\n");
57         RetCode=-6; //Si no pudo abrirse el archivo resultado.txt, RetCode toma el valor -6
58     }
59     if(number_file2==0)               //Si number_file2=0 se debe buscar y enviar el nombre
      del archivo '000'
60     {
61         flag2=0;                   //Bajada de la bandera número 2
62         system("find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/ -type f -
      name \'*000-*\' >/home/pi/Desktop/Archivos/telegrambot/telebot-
      master/test/test/resultado.txt"); //Búsqueda del archivo '000'
63         fgets(auxiliar2,150,name_file2); //Lectura del nombre del archivo del
      archivo resultado.txt
64         length=strlen(auxiliar2);        //Obtención de la longitud del nombre y la ruta del
      archivo '000'
65         i_parcial4=0;
66         while(auxiliar2[length-i_parcial4]!='/')
      //Se recorre el arreglo auxiliar desde el final hasta encontrar el carácter '/'
67         {   y se guarda el nombre del archivo en el arreglo auxiliar3
68             auxiliar3[i_parcial4]=auxiliar2[length-i_parcial4-2];
69             i_parcial4++;
70         }
71         length=strlen(auxiliar3); //Obtención de la longitud del nombre del archivo '000'
72         for(i_parcial4=0;i_parcial4<length-1;i_parcial4++)
      //Puesto que el arreglo auxiliar3 posee el nombre invertido del archivo,
73         {   debemos volver a invertirlo
74             aux_name2[i_parcial4]=auxiliar3[length-i_parcial4-2];
75         }
76         strcat(aux_name2,"\n\n");        //Concatenación de dos saltos de linea
77         strcat(aux_name2,auxiliar4);    //Concatenación del nombre del archivo con los
      nombres de los archivos anteriores

```

```

78 strcpy(auxiliar4,aux_name2);           //Copia del arreglo resultante anterior en el
                                         //arreglo auxiliar4
79 obtener_cantidad();                  //Obtención de la cantidad de archivos almacenables
80 cant_aux=cantidad_archivos;          //Asignación de la cantidad máxima a cant_aux
81 sprintf(auxiliar_cantidad,"%d",cantidad_archivos);
                                         //Conversión de la cantidad máxima de archivos a un arreglo de caracteres
82 strcat(cadena,auxiliar_cantidad);    //Concatenación del comando find, la ruta de
                                         //búsqueda y el número del archivo
83 strcat(cadena,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
                                         master/test/test/resultado.txt");      //Concatenación del arreglo
                                         //resultante anterior con el nombre del archivo resultado.txt
84 system(cadena);                   //Aplicación del comando
85 fseek(name_file2,0,SEEK_SET);       //Reubicación del puntero del archivo al inicio

86 char_aux=fgetc(name_file2);         //Obtención del primer carácter del archivo
87 if(char_aux=='/')                 //Si el primer carácter es '/' significa que existe un
                                         archivo con el nombre buscado
88 {
89     number_file2=cantidad_archivos; //number_file2 toma el valor numérico del
                                         mayor archivo numérico
90 }
91 else //Si no existe el archivo, significa que el último archivo a enviar es el 0
92 {           y ya se enviaron todos los archivos
93 if(i_parcial3!=9) //Si el archivo número 0 no es el último archivo a enviar
94 {
95 for(i_parcial5=0;i_parcial5<10;i_parcial5++) //Bucle for que recorre desde el
                                         //máximo número de archivo hasta la
                                         //cantidad máxima menos 10
96 {
97     char cadena2[]{"find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build/
                                         -type f -wholename \"*00\""}; //Cadena auxiliar 2
98     char aux_name3[400]={" "}; //Arreglo auxiliar para el nombre del archivo número 3
99     char auxiliar5[50]={" "}; //Arreglo auxiliar 5
100    cant_aux--;           //Disminución en 1 de la cantidad auxiliar
101    fclose(name_file2);   //Cierre del archivo resultado.txt para eliminar
                                         contenido anterior
102   name_file2=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
                                         master/test/test/resultado.txt","r+"); //Apertura del archivo resultado.txt
103   sprintf(auxiliar_cantidad,"%d",cant_aux); //Conversión de cant_aux
104   strcat(cadena2,auxiliar_cantidad); //Concatenación del comando, la ruta de
                                         //búsqueda y el número de archivo
105   strcat(cadena2,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
                                         master/test/test/resultado.txt"); //Concatenación del arreglo
                                         //resultante anterior con el nombre del archivo resultado.txt
106  system(cadena2);           //Aplicación del comando
107  fseek(name_file2,0,SEEK_SET); //Reubicación del puntero del archivo al inicio
108  char_aux=fgetc(name_file2); //Obtención del primer carácter
109 if(char_aux=='/') //Si el primer carácter es '/', existe el archivo buscado
110 {
111     fgets(auxiliar2,150,name_file2); //Obtención de la ruta y el nombre del archivo
112     length=strlen(auxiliar2);      //Longitud de la ruta y el nombre del archivo

```

```

113 i_parcial4=0;
114 while(auxiliar2[length-i_parcial4]!='/') //Obtención del nombre del archivo por
separado e invertido
115 {
116 auxiliar5[i_parcial4]=auxiliar2[length-i_parcial4-2];
117 i_parcial4++;
118 }
119 length=strlen(auxiliar5);
120 for(i_parcial4=0;i_parcial4<length-1;i_parcial4++) //Obtención del nombre del
archivo correcto
121 { aux_name3[i_parcial4]=auxiliar5[length-i_parcial4-2]; }
122 strcat(aux_name3,"\n\n"); //Concatenación del nombre con dos saltos de línea
123 strcat(aux_name3,auxiliar4); //Concatenación de los nombres obtenidos
anteriormente con el nuevo nombre
124 strcpy(auxiliar4,aux_name3); //Se copia el arreglo resultante anterior en el
arreglo auxiliar4
125 }
126 }
127 break; //Salida del bucle
128 }
129 }
130 }
131 else //Si number_file2 es distinto de 0 debe buscarse el siguiente archivo
132 { a enviar y armar el arreglo con los nombres de los 10 archivos buscados
133 sprintf(auxiliar_cantidad,"%d",number_file2); //Conversión de number_file2 a un
134 strcat(cadena,auxiliar_cantidad); //Concatenación del comando find, la ruta de
búsqueda y el número del archivo
135 strcat(cadena,"-*\"> /home/pi/Desktop/Archivos/telegrambot/telebot-
master/test/test/resultado.txt"); //Concatenación del
arreglo resultante anterior con el nombre del archivo resultado.txt
136 system(cadena); //Aplicación del comando
137 fseek(name_file2,0,SEEK_SET); //Reubicación del puntero del archivo al inicio
138 fgets(aux_name,150,name_file2); //Lectura del nombre del archivo
139 i_parcial4=0;
140 length=strlen(aux_name); //Obtención de la longitud del nombre y la ruta del archivo
141 while(aux_name[length-i_parcial4]!='d') //Se recorre el arreglo auxiliar desde
142 { el final hasta encontrar el carácter 'd' del directorio "build"
143 auxiliar3[i_parcial4]=aux_name[length-i_parcial4-1];
144 i_parcial4++;
145 }
146 length=strlen(auxiliar3); //Obtención de la longitud del nombre del archivo
147 i_parcial4=0;
148 while(auxiliar3[i_parcial4]!='/')
//Inversión del arreglo auxiliar3 para obtener el nombre correcto del archivo
149 {
150 aux_name2[i_parcial4]=auxiliar3[length-i_parcial4-2];
151 i_parcial4++;
152 }
153 strcat(aux_name2,"\n\n"); //Concatenación de dos saltos de linea
154 strcat(aux_name2,auxiliar4); //Concatenación del arreglo auxiliar4 con todos
los nombres ya buscados al nuevo nombre

```

```

155 strcpy(auxiliar4,aux_name2);           //Se copia el arreglo resultante anterior en
                                         //el arreglo auxiliar4
156 number_file2--;
157 }
158 fclose(name_file2);                  //Cierre del archivo resultado.txt
159 }
160 ret = telebot_send_message(handle, message.chat->id, auxiliar4, "HTML", false,
                           false, 0, ""); //Envío del arreglo con la lista de nombres de los 10 archivos
161 }

```

Explicación:

Debido a la complejidad del código se hará una pequeña explicación general de la función.

Se planteó el siguiente mecanismo. Mediante el uso de banderas y contadores, como en el caso del contador ‘lista’, cada vez que el usuario lista los 10 archivos anteriores este contador decrece en 1 y cada vez que se listan los 10 posteriores se incrementa en 1.

La primera estructura condicional en la línea 14 tiene en cuenta el caso de la primera llamada a la función de listar archivos. En este caso en el main se coloca este contador en -1, por lo que dentro de esta condición se obtienen variables como el último archivo creado, el más antiguo y el mayor numéricamente.

Luego comienza el bucle for principal. Este será el encargado de buscar los nombres de los 10 archivos, concatenar cada nombre del más antiguo al más nuevo en un solo arreglo general y enviar este arreglo al finalizar.

De la misma forma que en las dos funciones anteriores, la búsqueda de los archivos se guarda en el archivo “resultado.txt”. En el caso particular del archivo número 0, se busca el mismo en forma independiente. Ya que solo se envían los nombres de los archivos y no sus rutas, al leer del archivo resultado.txt, el cual contiene el nombre, la ruta y la fecha de creación, se obtiene por separado el nombre. Como la línea leída del archivo se lee desde el final hasta el comienzo del nombre del archivo, luego debe invertirse el arreglo para obtener el nombre correcto. Esto debe hacerse con todos los nombres de los archivos.

Además, en el caso de que se deba enviar el archivo 0, el siguiente nombre de archivo a buscar es el mayor numéricamente, solo si este existe. Esto se hace en la línea 87.

Cada nombre obtenido es concatenado con el anterior de forma que el listado se muestre desde el archivo más antiguo al más nuevo. Esto se realiza en las líneas 153 a 155.

Consideración: el código implementado desde la línea 93 a la línea 128 tiene en cuenta un caso poco probable pero el cual puede darse bajo ciertas condiciones. Para explicarlo se pondrá un ejemplo.

Suponga que inicialmente el parámetro de la cantidad de archivos almacenables era 23. Los archivos generados alcanzaron dicho número y el conteo se reinició, sobrescribiendo los primeros archivos hasta alcanzar el número 19. Posteriormente este parámetro se modificó a 25, por lo que ahora la cantidad máxima de archivos será 25 y no 23.

Lo que sucederá en esta situación es que se empezará listando desde el archivo 19 hacia archivos menores. Al llegar al archivo número 0, el siguiente nombre a buscar será el mayor numéricamente, es decir el 25. Pero este archivo no existe, por lo que se considerará que ya se enviaron todos los archivos y se saldrá de la función. Sin embargo, los archivos del número 23 al 20 incluido no se han enviado y si existen.

Para solucionar este problema se agregó este código cuya tarea es preguntar, aunque no exista el archivo número 25, si existen el archivo 24, el 23, el 22, y así sucesivamente, agregándolos al listado si estos existen. De esta forma se salva esta situación, con el detalle de que esta situación puede darse dentro de los últimos 10 archivos, por lo que se volverán a enviar los nombres de aquellos archivos que ya se enviaron y caigan dentro de estos últimos 10 números, como el caso de los archivos número 19 al 15.

Otra consideración es que, al igual que en las dos funciones anteriores, si por ejemplo, anteriormente el parámetro de cantidad de archivos almacenables era de 45 y existen estos archivos, y luego se modifica este parámetro a 35, el listado tendrá en cuenta los archivos del 0 al 35 y no se listarán del 36 al 45.

▪ Función mostrar_actuales:

El segundo menú de opciones que se le presenta al usuario posee opciones para la visualización y modificación de los parámetros de configuración. La primera función permite ver cuáles son los parámetros actuales junto con una serie de consideraciones que deben tenerse a la hora de modificarlos.

Diagrama de flujo:

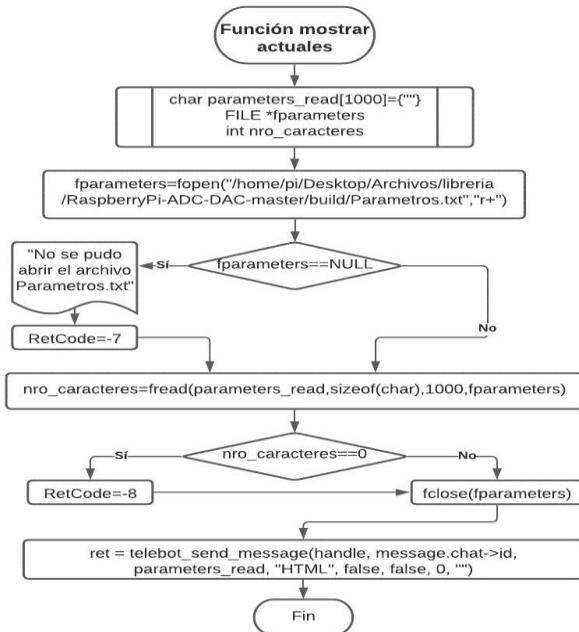


Figura 71: Diagrama de flujo de la función “mostrar_actuales”.

Código fuente:

```
/*/////////////////////////////Función "mostrar_actuales"////////////////////*/
1 void mostrar_actuales() //Función que muestra los parámetros de
                           //configuración actuales
2 {
3     char parameters_read[1000]={""}; //Arreglo donde se almacenarán los parámetros
4     FILE *fparameters; //Descriptor para el archivo Parametros.txt
5     int nro_caracteres; //Cantidad de caracteres leídos del archivo Parametros.txt
6     fparameters=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
                           DAC/build/Parametros.txt","r+"); //Apertura del archivo Parametros.txt
7     if(fparameters==NULL) //Si fparameters=NULL, no pudo abrirse el archivo
8     {
9         printf("\nNo se pudo abrir el archivo Parametros.txt\n");
10    RetCode=-7; //Si no pudo abrir el archivo Parametros.txt, RetCode toma el valor -7
11 }
12    nro_caracteres=fread(parameters_read,sizeof(char),1000,fparameters);
           //Lectura de los parámetros y almacenamiento en el arreglo parameters_read
13    if(nro_caracteres==0)
14    {
15        RetCode=-8; //Si la cantidad de caracteres leídos es 0, RetCode toma el valor -8
16    }
17    fclose(fparameters); //Cierre del archivo Parametros.txt
18    ret = telebot_send_message(handle, message.chat->id, parameters_read, "HTML",
           false, false, 0, ""); //Envío de los parámetros leídos
19 }
```

▪ Función modificar parámetros:

Esta función la única tarea que realiza es la de mostrar el listado de parámetros a modificar, pero no modifica los parámetros en sí. Se planteó de esta forma por dos razones.

La primera es que a pesar de existir menús jerárquicos (es decir, primero se presenta el menú con todas las opciones, luego el menú de los parámetros y por último la función), en la realización del código, cada menú y función implementada no requiere necesariamente que el usuario haya atravesado los menús anteriores. Por ejemplo, si desea modificar el parámetro frecuencia de muestreo, puede primero partir del menú principal, ir al menú de los parámetros, luego a la opción de modificar los parámetros y por último elegir el parámetro frecuencia. Sin embargo, si el menú de los parámetros a modificar ya se ha mostrado, el usuario puede directamente elegir cualquier otro parámetro para su modificación sin necesidad de volver a realizar el camino anteriormente descripto. Esto se pensó con el fin de agilizar y facilitar la interacción con el usuario. Por lo tanto, esta función sólo muestra el listado de parámetros con el propósito de darle al usuario independencia y capacidad de elegir otra opción si así lo desea.

La segunda razón de esta función es que la modificación de parámetros puede elegirse en base a seguir dos caminos: el primero es el ya descripto, colocando la opción “modificar parámetros”. El segundo es a través de la función “mostrar actuales”, ya que luego de mostrar los parámetros se le da la opción al usuario de modificarlos o no hacer nada con ellos.

Código fuente:

```
/*/////////////////////////////Función "modificar_parametros"////////////////////*/
1 void modificar_parametros()           //Función que presenta al usuario el listado
                                         de los parámetros a modificar
2 {
3     ret = telebot_send_message(handle, message.chat->id, "Elija el parametro a
                                         modificar:\n /Canal_de_disparo\n /Frecuencia_de_muestreo\n /Nivel_de_disparo\n
                                         /Muestras_post_trigger\n /Muestras_pre_trigger\n
                                         /Cantidad_archivos_almacenable\n" , "HTML", false, 0, "");
4 }
```

▪ Función modify_parameters(int flag_change):

Esta función es la encargada de efectivamente modificar los parámetros en el archivo de configuración.

Básicamente se definieron 6 arreglos que contienen las líneas principales escritas en el archivo de configuración. Cuando el usuario elige un parámetro para modificar, la función se queda esperando a que llegue un nuevo valor numérico (líneas 38 a 56). Una vez que el usuario envía este valor, se copia la línea correspondiente a un arreglo auxiliar, según la variable flag_change que representa el número de parámetro a modificar. Posteriormente se concatena el valor recibido con esta línea auxiliar, formando así la línea a escribir en el archivo (línea 118).

Diagrama de flujo:

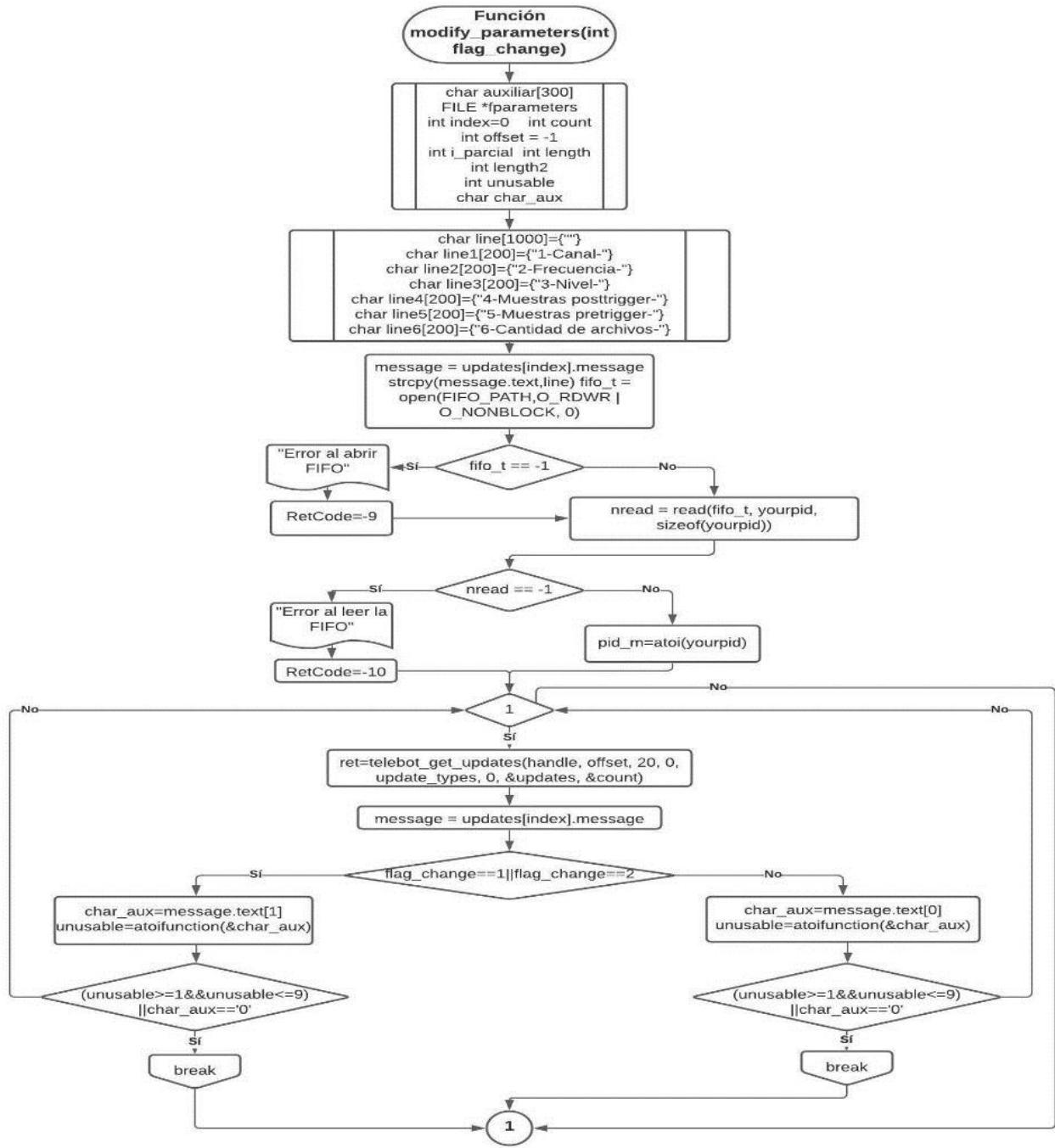




Figura 72: Diagrama de flujo de la función “modify_parameters”.

Código fuente:

```

/*
Función "modify_parameters"
1 void modify_parameters(int flag_change) //Función que modifica los parámetros en el archivo. Recibe como argumento la variable

```

```

    flag_change que representa al parámetro a
    modificar

2  {

3  char auxiliar[300];           //Arreglo auxiliar
4  FILE *fparameters;          //Descriptor para el archivo Parametros.txt
5  char line[1000]={""};        //Línea auxiliar general
6  char line1[200]={"1-Canal-"}; //1ra línea del archivo
7  char line2[200]={"2-Frecuencia-"}; //2da línea del archivo
8  char line3[200]={"3-Nivel-"};   //3ra línea del archivo
9  char line4[200]={"4-Muestras posttrigger-"}; //4ta línea del archivo
10 char line5[200]={"5-Muestras pretrigger-"}; //5ta línea del archivo
11 char line6[200]={"6-Cantidad de archivos-"}; //6ta línea del archivo
12 int index=0;                //Entero que nos permite, dentro del objeto que compone la
                                actualización, elegir el mensaje
13 int count;                  //Número de actualizaciones recibidas con #telebot_get_updates.
14 int offset = -1;            //Identificador de la primera actualización que se
                                devolverá. El desplazamiento negativo se puede especificar
                                para recuperar actualizaciones a partir de la actualización
                                de desplazamiento desde el final de la cola de
                                actualizaciones.

15 int i_parcial;             //Entero auxiliar
16 int length;                //Entero auxiliar con la longitud de un arreglo
17 int length2;               //2do entero auxiliar con la longitud del
arreglo
18 int unusable;              //Entero utilizado para la determinación del tipo de parámetro
19 char char_aux;              //Carácter auxiliar
20 message = updates[index].message; //Nuevo mensaje entrante de cualquier tipo:
                                texto, foto, sticker, etc.
21 strcpy(message.text,line); //Vaciamos el texto del mensaje, para evitar
                                cualquier posible carácter residual
22 fifo_t = open(FIFO_PATH,O_RDWR | O_NONBLOCK, 0); //Apertura de la FIFO en modo
                                lectura/escritura y no
                                bloqueante
23 if(fifo_t == -1)           //Si fifo_t=-1, ocurrió un error al intentar abrir la
FIFO
24 {
25 printf("\nError al abrir FIFO\n");
26 RetCode=-9;                 //Si no pudo abrirse la FIFO, RetCode toma el valor -9
27 }
28 nread = read(fifo_t, yourpid, sizeof(yourpid)); //Lectura de la FIFO
29 if(nread == -1)              //Si nread=-1, ocurrió un error al intentar leer de la
FIFO
30 {
31 printf("\nError al leer de la FIFO\n");
32 RetCode=-10;                 //Si se pudo leer de la FIFO, RetCode toma el valor -
10
33 }
34 else
35 {

```

```

36 pid_m=atoi(yourpid);           //Conversión numérica del PID del programa de
medición
37 }

38 while(1)
39 {
40     ret=telebot_get_updates(handle, offset, 20, 0, update_types, 0, &updates, &count);
        //Obtención de actualizaciones (mensajes entrantes)
41     message = updates[index].message;           //Asignación a message del nuevo mensaje
                                                 de texto entrante
42     if(flag_change==1||flag_change==2)          //Si se desea modificar el parámetro de
                                                 canal de disparo o frecuencia de
                                                 muestreo
43     {
44         char_aux=message.text[1];             //Asignación de la posición 1 del mensaje
                                                 entrante al carácter auxiliar
45         unusable=atoi(function(&char_aux)); //Conversión numérica del 2do carácter
                                                 recibido
46         if((unusable>=1&&unusable<=9)||char_aux=='0') //Si el carácter recibido es un
                                                 número, se sale del bucle
47     { break; }
48 }
49 else                           //Si se desea modificar alguno de los otros parámetros
50 {
51     char_aux=message.text[0];             //Asignación de la posición 0 del mensaje entrante
                                                 al carácter auxiliar
52     unusable=atoi(function(&char_aux)); //Conversión numérica del 1er carácter
                                                 recibido
53     if((unusable>=1&&unusable<=9)||char_aux=='0') //Si el carácter recibido es un
                                                 número, se sale del bucle
54     { break; }
55 }
56 }
57 fparameters=fopen("/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC
    /build/Parametros.txt","r+"); //Apertura del archivo Parametros.txt
58 if(fparameters==NULL) //Si fparameters=NULL, no pudo abrirse el archivo
    Parametros.txt
59 {
60     printf("\nNo se pudo abrir el archivo Parametros.txt\n");
61     RetCode=-11; //Si no pudo abrirse el archivo Parametros.txt, RetCode valdrá -11
62 }
63 switch(flag_change) //En base a la bandera flag_change que representa cada
    parámetro:
64 {
65     case 1:                      //Parámetro del canal de disparo
66     {
67         strcpy(line,line1);      //Copia de la linea 1 en la línea a copiar en el archivo
68         length=strlen(message.text); //Longitud del parámetro recibido
69         for(i_parcial=0;i_parcial<length;i_parcial++) //Corrimiento de un lugar hacia
                                                 la izquierda de los caracteres recibidos
                                                 para eliminar el carácter '/'
70     }

```

```

71     message.text[i_parcial]=message.text[i_parcial+1];
72 }
73 }
74 break;
75 case 2: //Parámetro de la frecuencia de muestreo
76 {
77     strcpy(line,line2); //Copia de la línea 2 en la línea a escribir
78     length=strlen(message.text); //Longitud del parámetro recibido
79     for(i_parcial=0;i_parcial<length;i_parcial++) //Corrimiento de un lugar hacia la
79                                izquierda de los caracteres recibidos
80                                para eliminar el carácter '/'
81     message.text[i_parcial]=message.text[i_parcial+1];
82 }
83 }
84 break;
85 case 3: //Parámetro del nivel de disparo
86 {
87     strcpy(line,line3); //Copia de la linea 3 en la línea a copiar en el archivo
88 }
89 break;
90 case 4: //Parámetro de la cantidad de muestras post disparo
91 {
92     strcpy(line,line4); //Copia de la linea 4 en la línea a copiar en el archivo
93 }
94 break;
95 case 5: //Parámetro de la cantidad de muestras pre disparo
96 {
97     strcpy(line,line5); //Copia de la linea 5 en la línea a copiar en el archivo
98 }
99 break;
100 case 6: //Parámetro de la cantidad de archivos almacenables
101 {
102     strcpy(line,line6); //Copia de la linea 6 en la línea a copiar en el archivo
103 }
104 break;
105 }
106 strcat(line,message.text); //Concatenación del parámetro recibido con la
106                                línea a copiar en el archivo
107 strcat(line,"\\n"); //Concatenación del salto de línea con la línea a
107                                copiar en el archivo
108 for(i_parcial=0;i_parcial<flag_change;i_parcial++)
108                                //Recorrido del archivo Parámetros.txt hasta llegar a la línea a modificar
109 { fgets(auxiliar,250,fparameters); }
110 length2=strlen(auxiliar); //Longitud de la línea actual a modificar
111 for(i_parcial=0;i_parcial<(14-flag_change);i_parcial++)
111                                //Recorrido del archivo Parámetros.txt desde el final de la línea a modificar
112                                hasta el final del mismo
113 fgets(auxiliar,300,fparameters);
114 length2=length2+strlen(auxiliar); //Cálculo de la cantidad de caracteres
recorridos

```

```

115 strcat(line,auxiliar);           //Concatenación de cada línea posterior a la
modificada
116 }
117 fseek(fparameters,-length2,SEEK_CUR);      //Reubicación del puntero del archivo
                                              hasta el principio de la línea
                                              modificada
118 fputs(line,fparameters);          //Escritura de los parámetros concatenados con
                                              la línea modificada en el archivo
119 fclose(fparameters);            //Cierre del archivo Parametros.txt
120 ret = telebot_send_message(handle, message.chat->id, "Aplicar los
cambios:\n\n/Aplicar_ahora.\n\n/Tal_vez_luego." , "HTML", false, false, 0, "");
                                              //Envío de las opciones posteriores al usuario
121 }

```

Consideraciones:

- La apertura y lectura de la FIFO se realiza dentro de esta función debido a que sólo se requerirá conocer el PID del proceso de medición y enviar una señal, si se modifica alguno de los parámetros. La FIFO debe abrirse en modo lectura/escritura (o especificar el modo O_NONBLOCK) para no ser bloqueante. Caso contrario, si el proceso de medición no ha iniciado, el proceso de comunicación remota se bloqueará hasta que el proceso de medición abra la FIFO en modo escritura o lectura/escritura.
- También, realizar la apertura de la FIFO en esta función trae como ventaja que, si por alguna razón, el programa de medición se reinicia y obtiene un nuevo PID, el proceso de comunicación obtendrá nuevamente su PID, permitiendo comunicarse con el nuevo proceso.
- Debido a que una vez dentro de esta función, se entra al bucle que espera a que llegue el valor del parámetro a modificar, se debe hacer una distinción respecto a si el nuevo mensaje recibido es una letra o un número, ya que todos los parámetros son número. Esto se hace en las líneas 46 y 53. Además, se debe tener en cuenta que para los dos primeros parámetros al usuario se le ofrece un listado de opciones, ya que tanto el canal de disparo, como la frecuencia de muestreo no pueden tomar cualquier valor. Es por esto que el 1er carácter recibido en estos casos es '/', y por lo tanto se deberá preguntar por el segundo carácter para determinar si es un número o no. Para los parámetros restantes, se pregunta por el primer carácter.
- Inicialmente, una vez obtenida la línea completa a escribir en el archivo, simplemente se optó por moverse dentro del archivo hasta la línea a modificar y sobrescribirla. Sin embargo, esto presentaba diversos problemas: se borraba el primer carácter de la línea siguiente, se concatenaba la línea modificada a la línea anterior, etc. La solución elegida a estos problemas fue que, previo a modificar la línea correspondiente, se realiza una copia de los parámetros posteriores junto con las recomendaciones en un arreglo auxiliar, se modifica la línea en cuestión y se vuelven a copiar en el archivo los parámetros posteriores almacenados en el arreglo auxiliar. Esto se realiza desde la línea 108 a la línea 118.

▪ Función main:

Declaración de variables:

```
/////////////////////////////Declaración de variables
locales/////////////////////
FILE *file_main;                                //Descriptor para el archivo resultado.txt
FILE *fp;                                         //Descriptor para el archivo token.txt
FILE *fm;                                         //Descriptor para el archivo menu.txt
FILE *ft;                                         //Descriptor para el archivo temperatura.txt
char name_file[400]={""};                         //Arreglo auxiliar para el nombre del archivo a
enviar
char comando[]={"vcgencmd measure_temp >
/home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test/temperatura.txt"};
//Arreglo con el comando que arroja la temperatura de la placa
char token[1024];                                //Arreglo auxiliar para la obtención del token del bot
char auxiliar[30];                                 //Arreglo auxiliar
char auxiliar2[200]={""};                          //2do arreglo auxiliar
char str[4096];                                   //Arreglo auxiliar con mensaje de bienvenida
int flag_parameters=0;                            //Variable bandera
int flag_change;                                  //Variable bandera
int index;                                       //Entero que nos permite, dentro del objeto que compone la
                                                 actualización, elegir el mensaje
int count;                                       //Número de actualizaciones recibidas con #telebot_get_updates.
int offset = -1;                                  //Identificador de la primera actualización que se devolverá.
                                                 El desplazamiento negativo se puede especificar para
                                                 recuperar actualizaciones a partir de la actualización de
                                                 desplazamiento desde el final de la cola de actualizaciones.
```

Creación e inicio del bot de Telegram:

Diagrama de flujo:

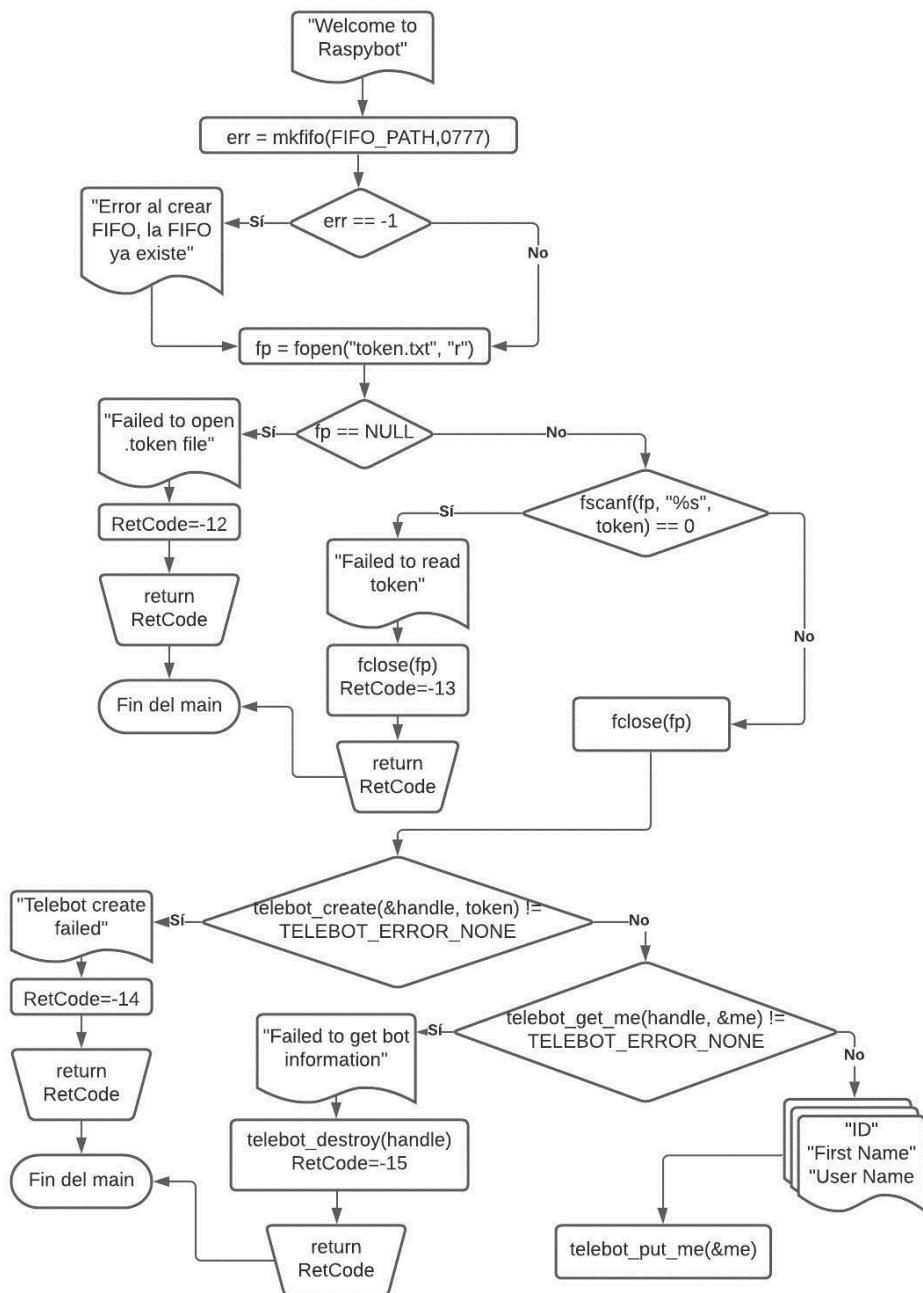


Figura 73: Diagrama de flujo del inicio del bot.

Código fuente:

```
///////////Creación e inicio del bot de Telegram///////////  
  
1 printf("Welcome to Raspybot\n");  
  
2 err = mkfifo(FIFO_PATH, 0777);      //Creación de la FIFO. En el 2do argumento se  
especifican los permisos de lectura, escritura y  
ejecución  
3 if(err == -1)                      //Si err=-1, no pudo crearse la FIFO  
4 { printf("\nError al crear FIFO, la FIFO ya existe\n"); }  
5 fp = fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-  
master/test/test/token.txt", "r");    //Apertura del archivo  
token.txt, el cual contiene el token del bot  
6 if (fp == NULL)                   //Si fp=NULL, no pudo abrirse el archivo token.txt  
7 {  
8     printf("Failed to open .token file\n");  
9     RetCode=-12; //Si no pudo abrirse el archivo token.txt, RetCode toma el valor -12  
10    return RetCode;                //Se retorna RetCode y termina el programa ya que la  
11 }      obtención del token es completamente necesaria para el funcionamiento del  
bot  
12 if (fscanf(fp, "%s", token) == 0) //Lectura del token.  
13 {  
14     printf("Failed to read token\n");  
15     fclose(fp);                  //Cierre del archivo token.txt  
16     RetCode=-13;                //Si no pudo leerse el archivo token.txt, RetCode toma el valor -  
13  
17     return RetCode;              //Se retorna RetCode y termina el programa ya que  
la  
18 }      obtención del token es completamente necesaria para el funcionamiento del  
bot  
19 printf("Token: %s\n", token);  
20 fclose(fp);  
21 if (telebot_create(&handle, token) != TELEBOT_ERROR_NONE) //Creación del bot de  
Telegram. Se le pasa como argumento un manejador y el token obtenido  
22 {  
23     printf("Telebot create failed\n");  
24     RetCode=-14;                  //Si no pudo crearse el bot, RetCode toma el valor  
-14  
25     return RetCode;              //Se retorna RetCode y termina el programa ya que la  
26 }      creación del bot es completamente necesaria para el funcionamiento del  
programa  
27 if (telebot_get_me(handle, &me) != TELEBOT_ERROR_NONE) //Obtención de la  
28 {      información del bot. Se le pasa como argumento el manejador y el objeto me,  
donde se obtendrán los datos del bot  
29     printf("Failed to get bot information\n");  
30     telebot_destroy(handle);  
31     RetCode=-15; //Si no pudo obtenerse información del bot, RetCode toma el valor -  
15  
32     return RetCode;              //Se retorna RetCode y termina el programa ya que la no  
33 }      obtención de información del bot es un indicador de una creación defectuosa
```

```

34 printf("ID: %d\n", me.id);
35 printf("First Name: %s\n", me.first_name);
36 printf("User Name: %s\n", me.username);
37 telebot_put_me(&me); //Esta función se utiliza para liberar la memoria
                        utilizada para la información obtenida sobre el propio bot
                        de telegram.

```

Bucle principal:

El bucle principal se divide en los menús y opciones presentadas al usuario.

Se tiene un bucle while principal en el cual el programa se mantendrá indefinidamente a menos que el usuario envíe el comando “/Exit”. Con la función ‘telebot_get_updates’ se reciben constantemente mensajes entrantes. El contador *count* indica cuantas actualizaciones (mensajes) se han recibido. Por lo tanto, solo entra al bucle for si ha llegado un mensaje. Caso contrario sigue monitoreando si llegan mensajes entrantes. Previo a salir del programa se debe eliminar el manejador del bot.

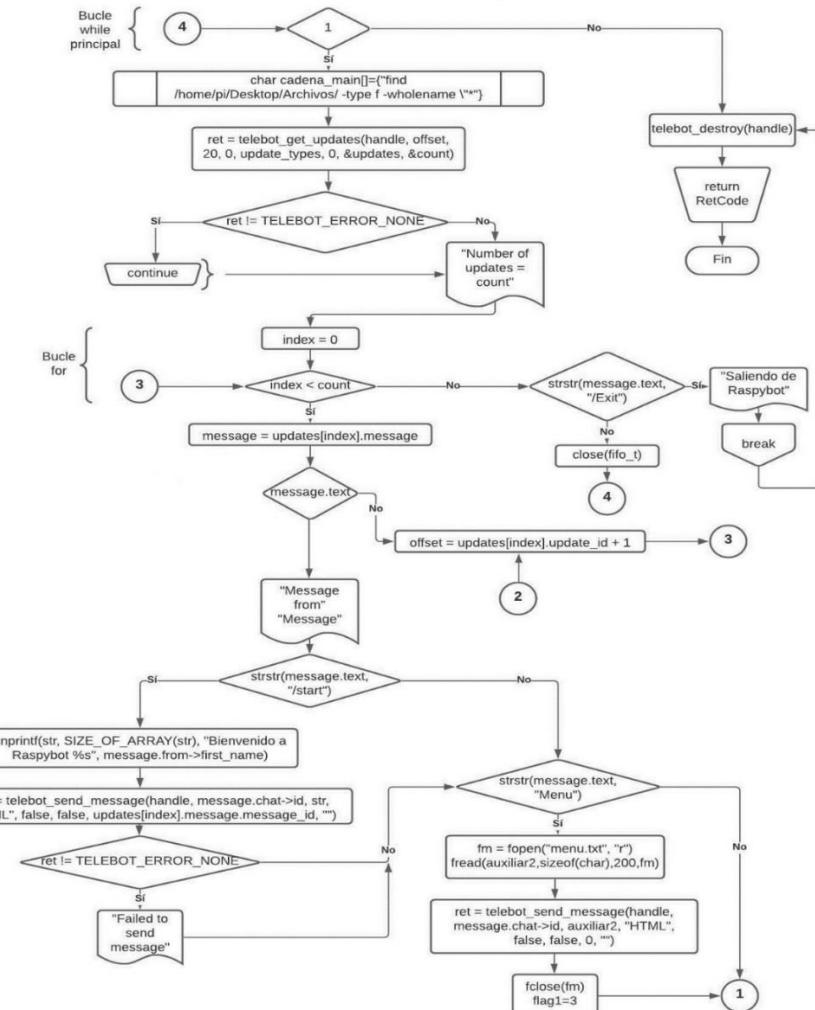


Figura 74: Diagrama de flujo del bucle principal.

El envío de acciones y la presentación de opciones se realiza mediante los llamados ‘comandos’. Su forma es el de ‘/comando’. No es más que un mensaje de texto el cual, al ser antecedido por una barra, es visualizado en color celeste en el chat del bot. La gran ventaja que presentan estos comandos es que actúan como un botón de texto, por lo que al tocar este comando se envía automáticamente el texto (véase “Experiencia”).

Por ejemplo, si el usuario envía el comando `/start`, el bot le retornará un mensaje de bienvenida. Por otro lado, si envía el mensaje “Menu” (sin acento), se lee el archivo menú.txt, el cual es un archivo de texto que contiene el menú principal. Este menú es enviado al chat en este caso.

```
/////////////////////////////Bucle principal////////////////////////////
1 while (1)
2 {
3     char cadena_main[]={ "find /home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-
    DAC/build/ -type f -wholename \ "*" };           //Arreglo con el comando y la ruta
    donde buscaremos el archivo
4     ret = telebot_get_updates(handle, offset, 20, 0, update_types, 0, &updates,
    &count); //Obtención de actualizaciones (mensajes entrantes)
5     if(ret != TELEBOT_ERROR_NONE) //Si ret es distinto de TELEBOT_ERROR_NONE se
        saltan
6     continue;   todas las instrucciones dentro del bucle hasta la siguiente iteración
7     printf("Number of updates: %d\n", count);
8     for (index = 0; index < count; index++)      //Se obtienen los mensajes entrantes
9     {                                              desde 0 hasta la cantidad de actualizaciones recibidas
10    message = updates[index].message;          //Dentro del objeto de la actualización
                                                entrante, obtenemos el mensaje recibido
11    if (message.text)                      //Si dentro del mensaje recibido se tiene texto:
12    {
13        printf("%s: %s \n", message.from->first_name, message.text);
14        if (strstr(message.text, "/start"))      //Si se recibe el comando /start, se
                                                    envía un mensaje de bienvenida
15        { snprintf(str, SIZE_OF_ARRAY(str), "Bienvenido a Raspybot %s", message.from-
            >first_name);
16        ret = telebot_send_message(handle, message.chat->id, str, "HTML", false, false,
            updates[index].message.message_id, "");
17        if (ret != TELEBOT_ERROR_NONE)
18        {
19            printf("Failed to send message: %d \n", ret);
20        }
21    }
/////////////////////////////Menú principal////////////////////////////
22 if (strstr(message.text, "Menu"))      //Si se recibe el mensaje "Menu", se
                                                presenta el menú principal al usuario
23 {
24     fm=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
        master/test/test/menu.txt", "r"); //Apertura del archivo menu.txt con
                                                el menú de opciones
```

```

25 fread(auxiliar2, sizeof(char), 200, fm);           //Lectura de las opciones
26 ret = telebot_send_message(handle, message.chat->id, auxiliar2, "HTML", false,
27   false, 0, ""); //Envío de las opciones al usuario
28 fclose(fm); //Cierre del archivo menu.txt
29 flag1=3; //Se reinicia la bandera 1

```

Menú de archivos:

La primera opción que se presenta en el menú principal está relacionada con la transferencia de archivos.

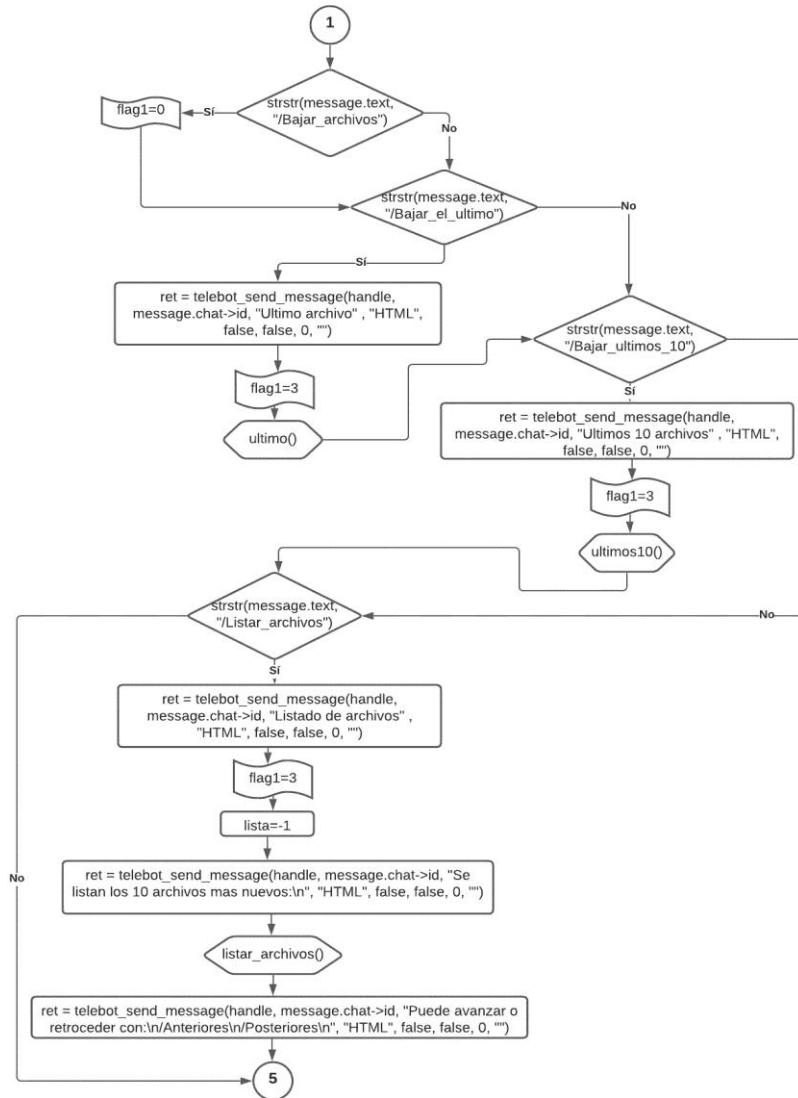


Figura 75: Diagrama de flujo del menú de archivos (1).

```

///////////////////////////////Descarga de archivos///////////////////////////////
1 if (stristr(message.text, "/Bajar_archivos")) //Si se recibe el comando
   /Bajar_archivos, la bandera 1 se coloca en 0

```

```

2 { flag1=0; }
3 if (strstr(message.text, "/Bajar_el_ultimo")) //Si se recibe el comando
4 {                               /Bajar_el_ultimo, se reinicia la bandera 1 y se llama a la
                                función que envía el último archivo
5 ret = telebot_send_message(handle, message.chat->id, "Ultimo archivo" , "HTML",
   false, false, 0, "");
6 flag1=3;
7 ultimo();
8 }
9 if (strstr(message.text, "/Bajar_ultimos_10")) //Si se recibe el comando
10 {                               /Bajar_ultimos_10, se reinicia la bandera 1 y se llama a la
                                función que envía los últimos 10 archivos
11 ret = telebot_send_message(handle, message.chat->id, "Ultimos 10 archivos" ,
   "HTML", false, false, 0, "");
12 flag1=3;
13 ultimos10();
14 }
15 if (strstr(message.text, "/Listar_archivos")) //Si se recibe el comando
16 {                               /Listar_archivos, se reinicia la bandera 1 y se llama a la
                                función que lista los archivos
17 ret = telebot_send_message(handle, message.chat->id, "Listado de archivos" ,
   "HTML", false, false, 0, "");
18 flag1=3;
19 lista=-1;           //La bandera lista toma el valor -1 indicando la situación inicial
20 ret = telebot_send_message(handle, message.chat->id, "Se listan los 10 archivos
   mas nuevos:\n", "HTML", false, false, 0, "");
21 listar_archivos();
22 ret = telebot_send_message(handle, message.chat->id, "Puede avanzar o retroceder
   con:\nAnteriores\nPosteriores\n", "HTML", false, false, 0, "");
23 }

```

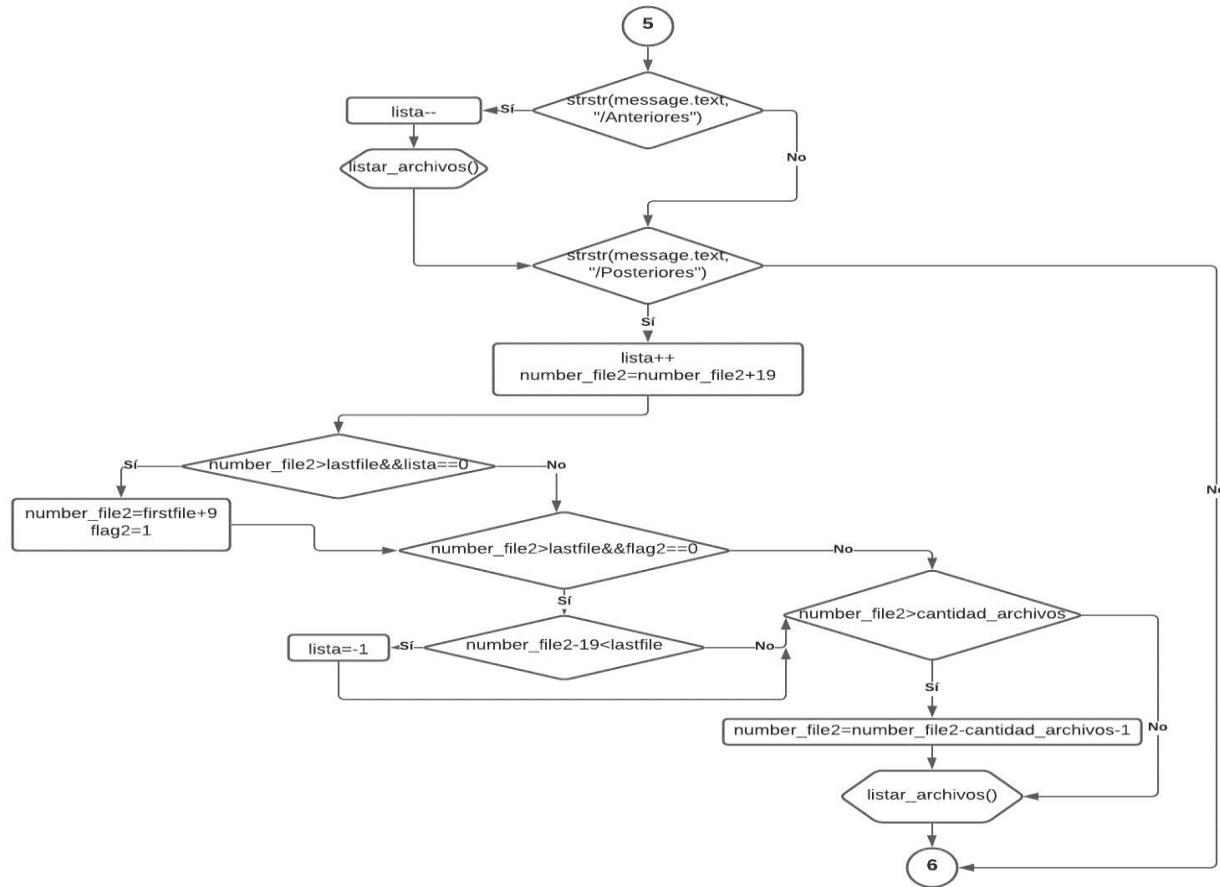


Figura 76: Diagrama de flujo del menú de archivos (2).

```

1  if (strstr(message.text, "/Anteriores")) //Si se recibe el comando /Anteriores,
2  { disminuye en 1 el valor de lista y se llama a la función que lista los archivos
3  lista--;
4  listar_archivos();
5 }
6  if (strstr(message.text, "/Posteriores")) //Si se recibe el comando /Anteriores,
7  { aumenta en 1 el valor de lista y se llama a la función que lista los archivos
8  lista++;
9  number_file2=number_file2+19; //El bucle for que obtiene el nombre de los
                                //archivos en listar_archivos finaliza con el
                                //número actual de archivo menos 9, por lo que
                                //debemos sumar 19 para listar los 10 posteriores
10 if(number_file2>lastfile&&lista==0)//Si se supera el número del archivo más nuevo
11 {
12     number_file2=firstfile+9;
13     flag2=1;
14 }
15 if(number_file2>lastfile&&flag2==0) //Si se supera el número del archivo más

```

```

16 {     nuevo y flag2=0, estamos en el caso en que ya se ha dado toda la vuelta al
17 if(number_file2-19<lastfile)          //Esta condición contempla el caso de que el
18 {     contador del número de archivo caiga entre los últimos 10 archivos
19     lista=-1;
20 }
21 }
22 if(number_file2>cantidad_archivos) //Si se supera la cantidad máxima de archivos,
23 {     number_file2 toma el valor de la diferencia con la cantidad máxima, menos 1
24 number_file2=number_file2-cantidad_archivos-1;
25 }
26 listar_archivos();
27 }

```

Uso de contadores y banderas:

El uso de contadores y banderas es fundamental para el control del programa y el reconocimiento de cada situación particular.

La variable **number_file2** es un contador que lleva la cuenta del número de archivo cuyo nombre debe ser buscado. Dentro del bucle for en la función **listar_archivos**, este contador disminuye en 1 cada vez que se busca un nuevo archivo.

Por otro lado, se tiene el contador de **lista**, el cual comienza en -1 al llamar por primera vez a la función **listar_archivos** y se decrementa en uno al listar los archivos anteriores y se incrementa en uno al listar los archivos posteriores. Es por ello, un contador que permite conocer cuantas veces se han listado archivos posteriores y cuantas veces se han listado archivos anteriores. Permite también reconocer, dentro de la función **listar_archivos**, la primera vez que la misma es llamada obteniendo todas las variables relevantes como el primer archivo, el último, el mayor numéricamente, etc.

Si el usuario envía el comando **/Anteriores** lo único que debe hacerse es decrementar el contador de **lista**, ya que es el propio lazo for dentro de la función es el que decremente el contador de archivos **number_file2**. Caso contrario, si el usuario envía el comando **/Posteriores**, además de incrementar el contador de **lista** se deben sumar 19 al contador de archivo. Esto se debe a que se ha supuesto que antes de enviar el comando **/Posteriores** se ha llamado al menos una vez a la función de **listar_archivos**, ya que es el camino lógico desde el menú principal. Por lo tanto, el contador de archivos queda en:

$$\text{number_file2} = \text{número actual} - 9$$

Se restan -9 en el lazo for ya que se cuenta también el archivo más nuevo. Por esto, si se desean listar los 10 archivos posteriores utilizando el contador de archivos, a este se le deben sumar 19 y no 10; de esta forma queda:

$$\text{number_file2} = (\text{número actual} - 9) + 19 = \text{número actual} + 10$$

Con número actual se refiere al archivo más nuevo de cada conjunto de 10 archivos enviados. Al listar archivos se pueden presentar 3 situaciones:

1. Listar los 10 archivos más nuevos por primera vez y luego listar los 10 archivos posteriores. En este caso el contador de **lista** valdrá 0, el archivo más antiguo **firstfile** se incrementa en 9 y se levanta la bandera 2 indicando esta situación. Esta condición se evalúa en la línea 10.
2. El segundo caso se da cuando se listan sucesivamente los archivos posteriores hasta llegar al punto de comenzar nuevamente en el punto inicial, por lo que se ha recorrido el listado total de archivos. En este caso se debe reiniciar el conteo de lista y bajar la bandera 2. Esta condición contempla el caso de que el conteo de archivos quede nuevamente entre los 10 archivos más nuevos. Reiniciar el conteo de lista evita que se comiencen a producir corrimientos en los 10 archivos mostrados en cada envío de los comandos **/Anteriores** y **/Posteriores** cuando la cantidad total de archivos no es un múltiplo de 10. Este segundo caso se evalúa en las líneas 15 y 17.
3. El tercer caso se da cuando se supera la cantidad máxima de archivos. Para ello se realiza la siguiente cuenta, antes de listar los archivos:

$$\text{number_file2} = (\text{número actual} + 19) - \text{cantidad máxima de archivos}$$

De esta forma, si por ejemplo el número actual de archivo más 19 es igual a 205 y la cantidad máxima de archivos es 200, entonces se realiza la diferencia y el contador de archivos toma el valor 5, listando primeramente del archivo número 5 al 0, y luego del 200 al 196.

Esta condición se evalúa en la línea 22.

Elección del archivo a enviar:

El listado de archivos presentado le da la posibilidad al usuario de elegir específicamente el archivo a descargar.

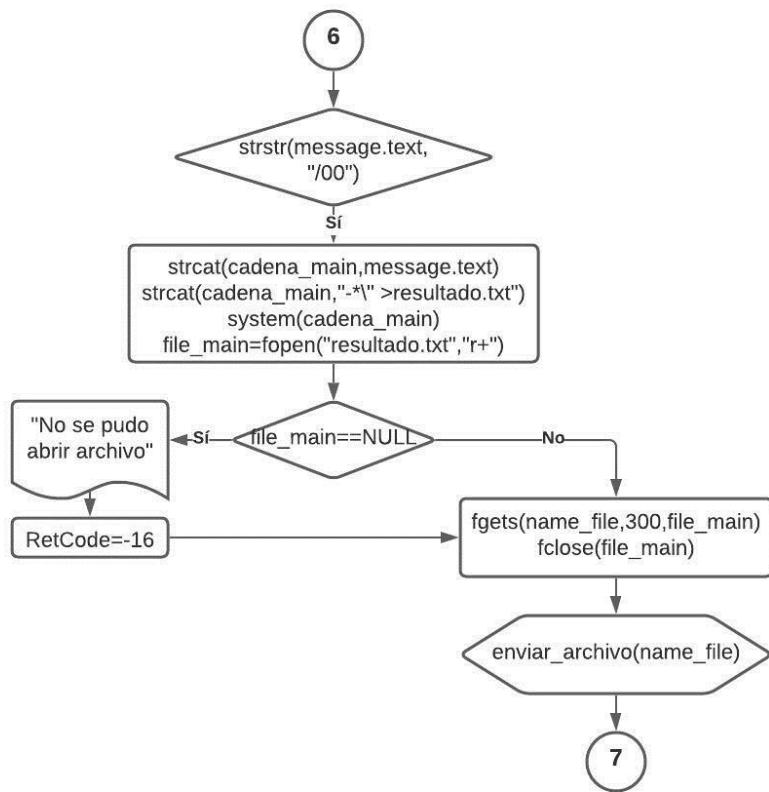


Figura 77: Diagrama de flujo del envío de archivos particulares.

```

1 if (strstr(message.text, "/00"))           //Si el mensaje recibido contiene "/00", se
debe
2 {                                         enviar el archivo cuyo número envió el usuario
3 strcat(cadena_main,message.text);        //Concatenación con el comando, la ruta de
                                           búsqueda y el número recibido
4 strcat(cadena_main,"-*\" >/home/pi/Desktop/Archivos/telegrambot/telebot-
master/test/test/resultado.txt"); //Concatenación del arreglo
                                           resultante anterior con el archivo donde guardaremos el
                                           resultado
5 system(cadena_main);                   //Aplicación del comando de búsqueda
6 file_main=fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
master/test/test/resultado.txt","r+");   //Apertura del
                                           archivo resultado.txt
7 if(file_main==NULL) //Si file_main=NULL, no pudo abrirse el archivo resultado.txt
8 {
9 printf("No se pudo abrir archivo\n");
10 RetCode=-16;                         //Si no pudo abrirse el archivo resultado.txt,
                                           RetCode toma el valor -16
11 }
12 fgets(name_file,300,file_main);       //Obtención de la ruta y el nombre del archivo
13 fclose(file_main);                  //Cierre del archivo resultado.txt
14 enviar_archivo(name_file);         //Llamada a la función que envía el archivo, con
                                           la ruta y el nombre del archivo como argumento

```

Si el mensaje recibido contiene la cadena “/00”, significa que el usuario requiere la descarga de un archivo en particular pues recuerde que todos los archivos comienzan con ‘00’.

Modificación de parámetros:



Figura 78: Diagrama de flujo de la modificación de parámetros.

La bandera **flag_parameters** permite discriminar si el comando **/Si** y **/No** han sido enviados luego de mostrar los parámetros actuales y por lo tanto se está indicando que efectivamente se desea modificar o no los parámetros de configuración.

La variable **flag_change** representa el número de parámetro a modificar y recuerde que es utilizada por la función **modify_parameters** para determinar la línea a sobrescribir en el archivo de configuración.

```
1 if (strstr(message.text, "/Parametros")) //Si se recibe el comando /Parametros, la
2 { flag1=1; }
3 if (strstr(message.text, "/Mostrar_actuales")) //Si se recibe el comando
4 {     /Mostrar_actuales, se reinicia la bandera 1, se llama a la función que
      muestra los parámetros actuales y se levanta la bandera flag_parameters
5 ret = telebot_send_message(handle, message.chat->id, "Se muestran los parametros
      actuales", "HTML", false, false, 0, "");
6 flag1=3;
7 mostrar_actuales();
8 ret = telebot_send_message(handle, message.chat->id, "\nDesea modificar alguno de
      los parametros:\n/Si\n/No", "HTML", false, false, 0, "");
9 flag_parameters=1;
10 }
11 if (strstr(message.text,"/Si") && flag_parameters==1) //Si se recibe el comando /Si
12 {     y además flag_parameters se encuentra en alto, se llama a la función que lista
          los comandos para modificar parametros y se baja la bandera flag_parameters
13 modificar_parametros();
14 flag_parameters=0;
15 }
16 if (strstr(message.text,"/No") && flag_parameters==1) //Si se recibe el comando /No
17 {     y además flag_parameters se encuentra en alto, no se realiza ninguna
          acción y se baja la bandera flag_parameters
18 flag_parameters=0;
19 }
20 if (strstr(message.text, "/Modificar_parametros")) //Si se recibe el comando
21 {     /Modificar_parametros, se reinicia la bandera 1 y se llama a la función
          que muestra la lista de parámetros al usuario
22 flag1=3;
23 modificar_parametros();
24 }
25 if (strstr(message.text, "/Canal_de_disparo")) //Si se recibe el comando
26 {     /Canal_de_disparo, se muestran los números de los canales
          y se llama a la función modify_parameters
27 ret=telebot_send_message(handle,message.chat->id,"Elija el canal de
      disparo:\n/0\n/1\n/2\n/3\n/4\n/5\n/6\n/7", "HTML", false, false,
      0, "");
28 flag_change=1;
29 modify_parameters(flag_change);
30 }
31 if (strstr(message.text, "/Frecuencia_de_muestreo")) //Si se recibe el comando
32 {     /Frecuencia_de_muestreo, se muestran las frecuencias de muestreo
          seleccionables y se llama a la función modify_parameters
```

```

33 ret = telebot_send_message(handle, message.chat->id,"Ingrese la frecuencia de
muestreo en muestras por segundo:\n/30000\n\n/15000\n\n/7500\n\n/3750\n\n/2000\n\n
/1000\n\n/500\n\n/100\n\n/60\n\n/50\n\n/30\n\n/25\n\n/15\n\n/10\n\n/5\n\n/2\n\n",
"HTML", false, false, 0, "");
34 flag_change=2;
35 modify_parameters(flag_change);
36 }
37 if (strstr(message.text, "/Nivel_de_disparo")) //Si se recibe el comando
38 { /Nivel_de_disparo, se muestra el formato del número a ingresar y se llama
a la función modify_parameters
39 ret = telebot_send_message(handle, message.chat->id,"Ingrese el nivel de disparo en
el formato \"U.dc\". Por ejemplo, \"1.38\", en volts.\n", "HTML", false, false, 0,
(""));
40 flag_change=3;
41 modify_parameters(flag_change);
42 }
43 if (strstr(message.text, "/Muestras_post_trigger")) //Si se recibe el comando
44 { /Muestras_post_trigger, se muestra la cantidad máxima a ingresar y se
llama a la función modify_parameters
45 ret = telebot_send_message(handle, message.chat->id,"Ingrese la cantidad de
muestras post trigger. *Tenga en cuenta que la suma de la cantidad de muestras pre
y post disparo no deben superar las 39000 muestras.*", "HTML", false, false, 0,
(""));
46 flag_change=4;
47 modify_parameters(flag_change);
48 }
49 if (strstr(message.text, "/Muestras_pre_trigger")) //Si se recibe el comando
50 { /Muestras_pre_trigger, se muestra la cantidad máxima a ingresar y se llama
a la función modify_parameters
51 ret = telebot_send_message(handle, message.chat->id,"Ingrese la cantidad de
muestras pre trigger. *Tenga en cuenta que la suma de la cantidad de muestras pre
y post disparo no deben superar las 39000 muestras.*", "HTML", false, false, 0,
(""));
52 flag_change=5;
53 modify_parameters(flag_change);
54 }
55 if (strstr(message.text, "/Cantidad_archivos_almacenables")) //Si se recibe el
56 { comando /Cantidad_archivos_almacenables, se muestra la consideración a
tener en cuenta y se llama a la función modify_parameters
57 ret = telebot_send_message(handle, message.chat->id,"Ingrese la cantidad de
archivos almacenables. *Una vez alcanzada la cantidad especificada, los nuevos
archivos sobrescribirán a los más antiguos.* ", "HTML", false, false, 0, "");
58 flag_change=6;
59 modify_parameters(flag_change);
60 }

```

Aplicación de parámetros y menús secundarios:

Recuerde que en el programa de medición los cambios en los parámetros de configuración no son aplicados hasta que arribe una señal proveniente del programa de comunicación remoto.

Si el usuario decide aplicar los cambios, se envía una señal al proceso cuyo PID se obtuvo de la FIFO.

Por otro lado, tanto la opción de bajar archivos como de modificar los parámetros en el menú principal modifican la bandera 1. Esta bandera es utilizada para presentar, según su valor, el menú de transferencia de archivos o el menú de modificación de parámetros. Una vez seleccionada alguna de las opciones de estos menús, la bandera 1 vuelve a su estado original para dejar de mostrar estos menús.

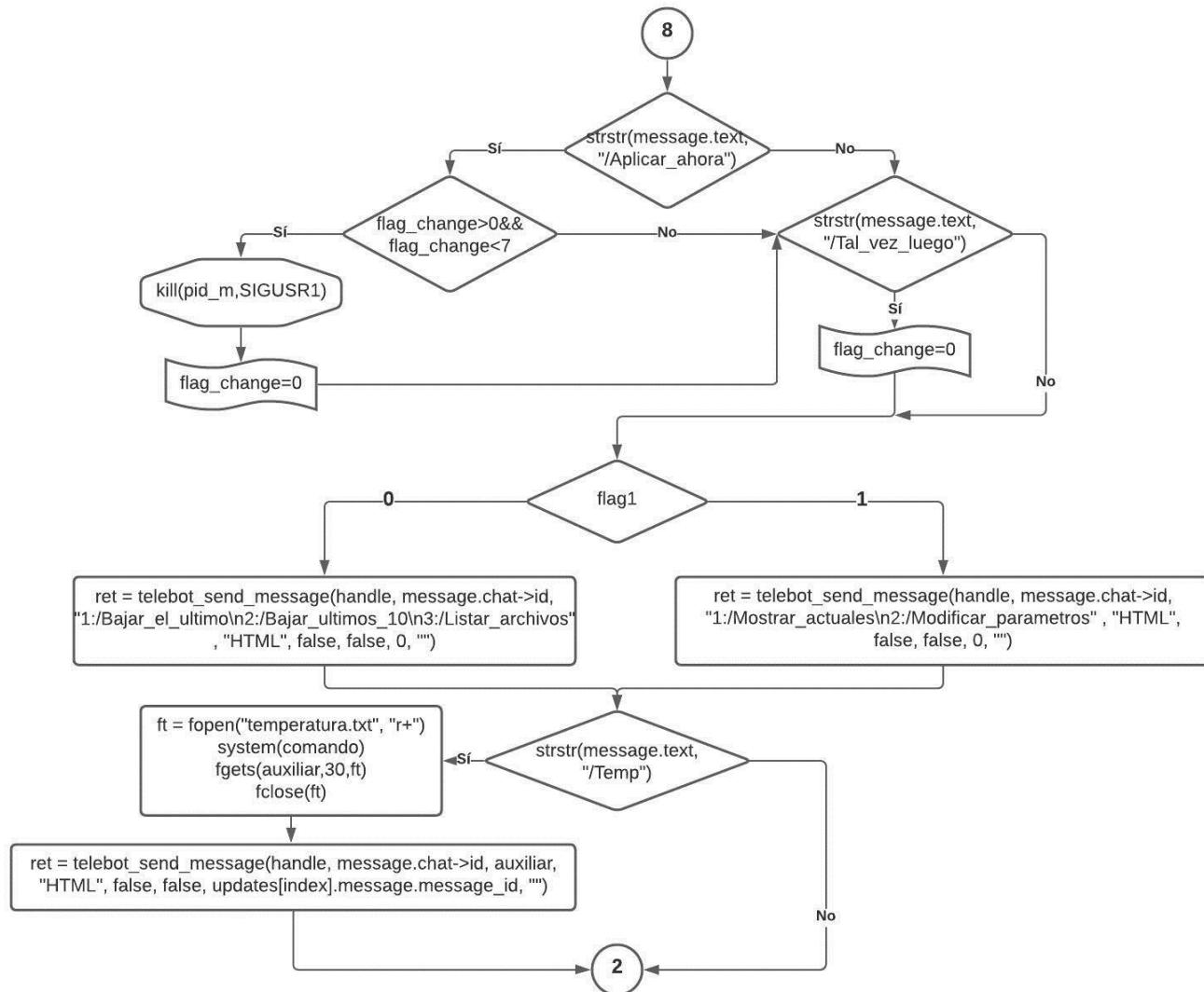


Figura 79: Diagrama de flujo de la aplicación de parámetros y menús secundarios.

```

1 if (strstr(message.text, "/Aplicar_ahora")) //Si se recibe el comando
2 {   /Aplicar_ahora y la variable flag_change se encuentra entre 0 y 7, indicando
     que se ha modificado algún parámetro:
3   if(flag_change>0&&flag_change<7)
4   {

```

```

5 kill(pid_m,SIGUSR1); //Se envía la señal SIGUSR1 al proceso cuyo PID recibimos a
6 flag_change=0;           través de la FIFO, en este caso del programa de medición.
                           La bandera flag_change toma el valor 0
7 }
8 }
9 if (strstr(message.text, "/Tal_vez_luego")) //Si se recibe el comando
10 { /Tal_vez_luego, simplemente se coloca en 0 la bandera flag_change
11 flag_change=0;
12 switch(flag1)           //En base a la bandera 1, se muestran las opciones para la
13 {                         descarga de archivos o para la modificación de parámetros
14 case 0:
15 {
16 ret = telebot_send_message(handle, message.chat->id, "1:/Bajar_el_ultimo\n"
   2:/Bajar_ultimos_10\n3:/Listar_archivos" , "HTML", false, false, 0, "");
17 };
18 break;
19 case 1:
20 {
21 ret = telebot_send_message(handle, message.chat->id, "1:/Mostrar_actuales\n"
   2:/Modificar_parametros" , "HTML", false, false, 0, "");
22 };
23 break;
24 }
25 if (strstr(message.text, "/Temp")) //Si se recibe el comando /Temp, se aplica el
26 { comando que permite obtener la temperatura de la placa, se lee la temperatura
   medida del archivo temperatura.txt y se envía al usuario
27 ft = fopen("/home/pi/Desktop/Archivos/telegrambot/telebot-
   master/test/test/temperatura.txt", "r+");
28 system(comando);
29 fgets(auxiliar,30,ft);
30 ret = telebot_send_message(handle, message.chat->id, auxiliar, "HTML", false,
   false, updates[index].message.message_id, "");
31 fclose(ft);
32 }
33 if (strstr(message.text, "/dice")) //Si se recibe el comando /dice, se envía al
   usuario una imagen descargada
34 {
35 telebot_send_dice(handle, message.chat->id, false, 0, "");
36 char image[]={"/home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test
   /download.jpeg"};
37 ret=telebot_send_photo(handle, message.chat->id,image,true,"","","",false,
   updates[index].message.message_id,"");
38 if (ret != TELEBOT_ERROR_NONE)
39 { printf("Failed to send photo: %d \n", ret); }
40 }
41 }
42 offset = updates[index].update_id + 1; //Identificador único de la actualización.
43 } Los identificadores de actualización comienzan con un cierto número positivo
   y aumentan secuencialmente.
44 if (strstr(message.text, "/Exit")) //Si se recibe el comando /Exit, se sale del
   bucle principal y se termina el programa

```

```

45 {
46 printf("\nSaliendo de Raspybot\n");
47 break;
48 }
49 close(fifo_t); //Cierre del descriptor de archivo de la FIFO
50 }
51 telebot_destroy(handle); //Se destruye el manejador del bot
52 return RetCode;
53 }

```

Listado de errores:

- **RetCode = 0** : El programa transcurrió sin errores.
- **RetCode = -1** : No pudo abrirse el archivo **Parametros.txt** en la función **Obtener_cantidad()**.
- **RetCode = -2** : No pudo abrirse el archivo **lastfile.txt** en la función **ultimo()**.
- **RetCode = -3** : No pudo abrirse el archivo **resultado.txt** en la función **ultimo()**.
- **RetCode = -4** : No pudo abrirse el archivo **lastfile.txt** o **resultados.txt** en la función **ultimos10()**.
- **RetCode = -5** : No pudo abrirse el archivo **lastfile.txt** en la función **listar_archivos()**.
- **RetCode = -6** : No pudo abrirse el archivo **resultado.txt** en la función **listar_archivos()**.
- **RetCode = -7** : No pudo abrirse el archivo **Parametros.txt** en la función **mostrar_actuales()**.
- **RetCode = -8** : Si la cantidad de caracteres leídos del archivo **Parametros.txt** es 0 en la función **mostrar_actuales()**.
- **RetCode = -9** : Si no pudo abrirse la FIFO en la función **modify_parameters**.
- **RetCode = -10** : Si no se pudo leer de la FIFO en la función **modify_parameters**.
- **RetCode = -11** : No pudo abrirse el archivo **Parametros.txt** en la función **modify_parameters**.
- **RetCode = -12** : Si no pudo abrirse el archivo **token.txt** en el main.
- **RetCode = -13** : Si no pudo leerse el archivo **token.txt** en el main.
- **RetCode = -14** : Si no pudo crearse el bot en el main.
- **RetCode = -15** : Si no pudo obtenerse información del bot.

- RetCode = -16 : Si no pudo abrirse el archivo resultado.txt.

7.3 TRASNFERENCIA DE ARCHIVOS A SERVIDOR FTP

Una última alternativa para la transferencia remota de archivos fue implementada en caso de que en un futuro sea requerida. Esta alternativa consta del uso de un servidor FTP, que como sus siglas lo indican, utiliza un Protocolo de transferencia de archivos, el cual es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

Solo se explicarán los pasos en forma breve para conectarse, subir y descargar archivos a un servidor FTP desde la terminal de comandos.

Fue de especial ayuda los aportes brindados por el Ing. Sebastián Tobar, quien además de brindar los conocimientos necesarios proporcionó como servidor de prueba las placas EDUCIAA del Laboratorio Remoto para Sistemas Embebidos del laboratorio Gridtics en la UTN Facultad Regional Mendoza.

Se utilizó SFTP (**SSH File Transfer Protocol** o **Secure File Transfer Protocol**) que es un protocolo del nivel de aplicación que proporciona la funcionalidad necesaria para la transferencia y manipulación de archivos sobre un flujo de datos fiable. Se utiliza comúnmente con SSH para proporcionar la seguridad a los datos, aunque permite ser usado con otros protocolos de seguridad. Por lo tanto, la seguridad no la provee directamente el protocolo SFTP, sino SSH o el protocolo que sea utilizado en su caso para este cometido.

Para conectarse al servidor se debe ejecutar el siguiente comando:

```
sftp -oPort=[PUERTO] [USUARIO]@[DIRECCIÓN IP]
```

```
pi@raspberrypi:~ $ sftp -oPort=  fernando.guerrero@[REDACTED]
The authenticity of host '[REDACTED]: ([REDACTED]): ' can't be established.
ECDSA key fingerprint is SHA256:nu8f/yqHyakgAE9199mEcw51DesfiChGqXmg1OFBwB4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[REDACTED]:' (ECDSA) to the list of known hosts.
fernando.guerrero@[REDACTED]'s password: [REDACTED]
Connected to fernando.guerrero@[REDACTED].
sftp> ls
firmware_v3
```

Figura 80: Conexión a servidor FTP remoto.

El Puerto y la dirección IP no se muestran por cuestiones de privacidad.

En la autenticación del usuario se solicitará su contraseña y una vez confirmada se encontrará conectado al servidor.

Para la descarga de archivos ejecutar:

```
get [NOMBRE DEL ARCHIVO] [DIRECTORIO LOCAL] [DIRECTORIO REMOTO]
```

```

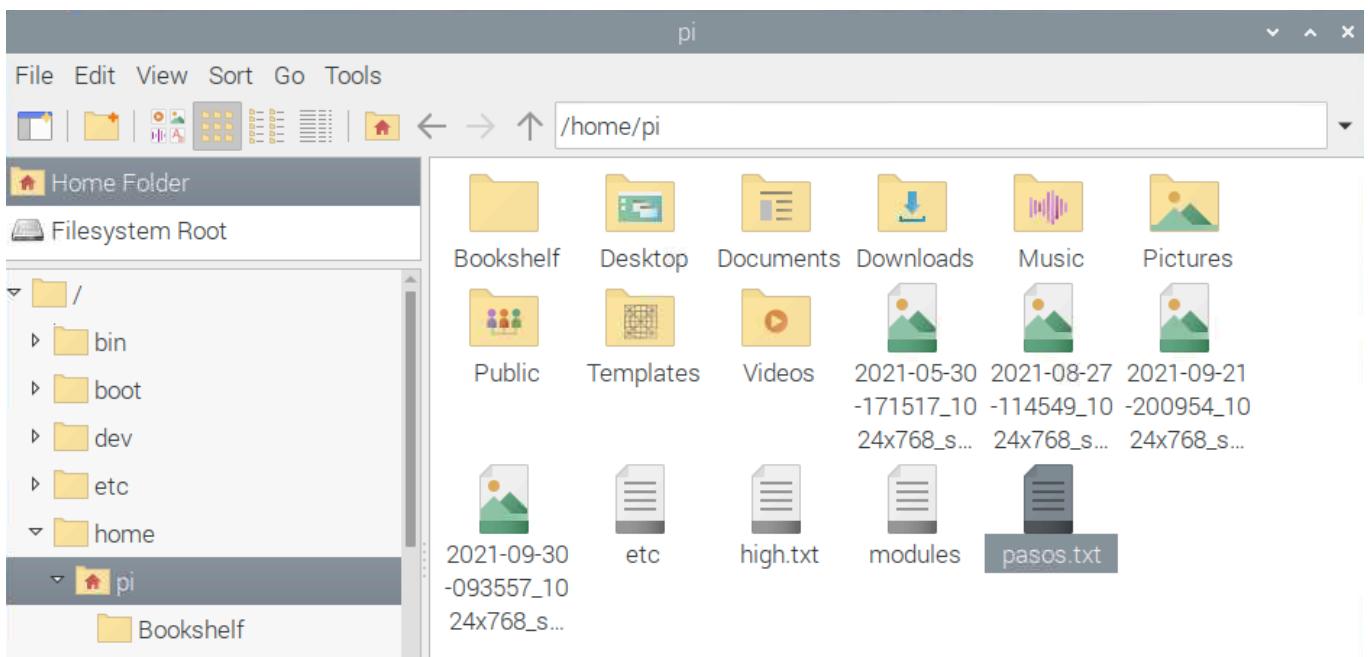
sftp> ls
ejercicio-14-01           ejercicio1-freeRTOS      ejercicio2-freeRTOS
pasos.txt                  tp14                   tp15
sftp> get pasos.txt /home/pi  /home/fernando.guerrero/firmware_v3/ejercicios/
Fetching /home/fernando.guerrero/firmware_v3/ejercicios/pasos.txt to /home/pi/pa
sos.txt
/home/fernando.guerrero/firmware_v3/ejercicio 100% 945      4.8KB/s  00:00

```

Figura 81: Descarga de archivos de servidor FTP remoto.

En este caso se transferirá el archivo *pasos.txt* al directorio local */home/pi* desde el directorio remoto */home/fernando.guerrero/firmware_v3/ejercicios*.

Y puede observar que el archivo *pasos.txt* fue transferido exitosamente a nuestro ordenador Raspberry Pi.



Para los fines de nuestro dispositivo, puede implementarse que cada vez que se genere un nuevo archivo, la Raspberry Pi lo suba a un servidor remoto del CONICET. Esto puede hacerse a través del siguiente comando:

put [DIRECTORIO LOCAL/NOMBRE DEL ARCHIVO] [DIRECTORIO REMOTO]

```

sftp> put /home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test/doc.txt
      /home/fernando.guerrero/firmware_v3/
Uploading /home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test/doc.txt
      to /home/fernando.guerrero/firmware_v3/doc.txt
/home/pi/Desktop/Archivos/telegrambot/telebot 100%   28      0.7KB/s  00:00
sftp> ls
firmware_v3
sftp> cd firmware_v3/
sftp> ls
LICENSE           Makefile          README.md        doc.txt          documentation
ejercicios        examples         libs             program.mk     scripts
templates         test

```

Figura 81: Subida de archivos a servidor FTP remoto.

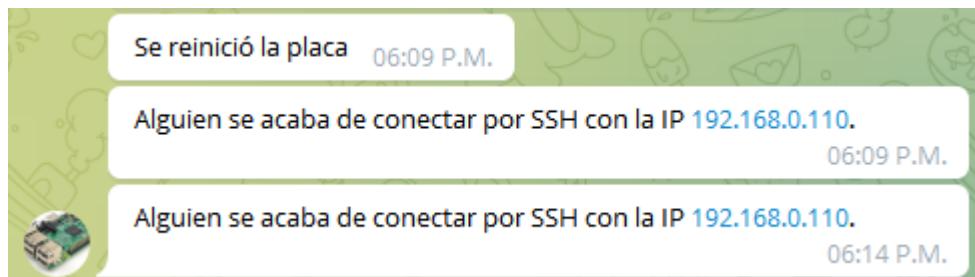
De esta forma se transfirió el archivo *doc.txt* en el directorio local */home/pi/Desktop/telegrambot/telebot-master/test/test* al directorio remoto */home/fernando.guerrero/firmware_v3/* y puede observar que efectivamente fue transferido el documento *doc.txt*.

Este es el método a utilizar si se desea la transferencia de archivos remota a través de un servidor FTP.

7.4 EXPERIENCIA

Una vez visto la lógica y el código desarrollado, puede describirse la experiencia que tendrá el usuario y las opciones presentadas.

Al iniciar la Raspberry Pi aparecerán los dos mensajes configurados: el de reinicio de la Raspberry Pi y de la conexión SSH.



Aparecerán los mensajes de que se ha realizado una conexión SSH cada vez que ingrese a través de la terminal, o desde el programa PuTTY, indicando la dirección IP del usuario conectado.

Si envía el comando **/start** recibirá un mensaje de bienvenida:

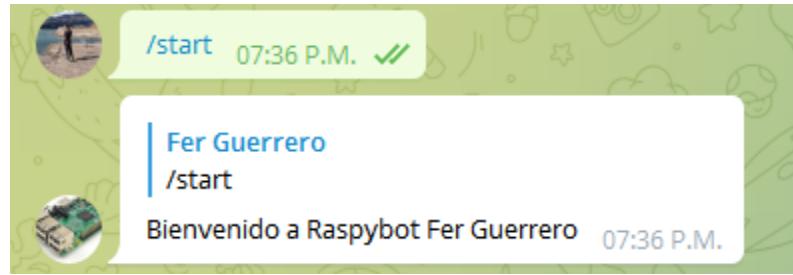


Figura 82: Mensaje de bienvenida del bot.

- **Menú principal:**

Para acceder al menú principal debe enviar la palabra “Menu” (sin acento):

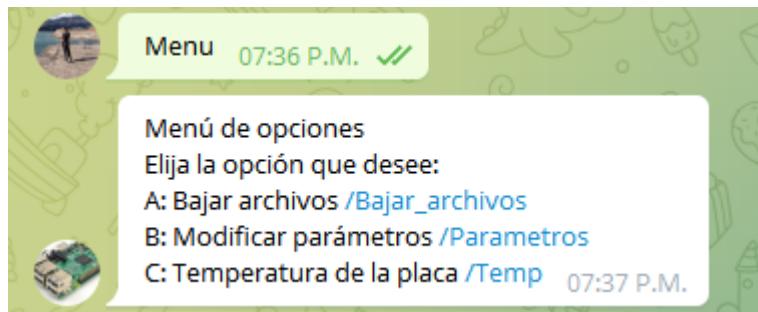


Figura 83: Menú principal de opciones.

Evite el uso de acentos debido a que suelen causar errores en el envío de mensajes. Note que los comandos se muestran en color celeste. Puede cliquear sobre estos comandos como si de un botón se tratara e inmediatamente se enviará el comando correspondiente.

- **Opción A: Bajar archivos**

Al elegir la opción A para la descarga de archivos, se presenta el siguiente submenú:



Figura 84: Menú de archivos.

- **Bajar el último archivo generado:**

Para descargar el archivo más nuevo generado, elija la opción número 1 a través del comando **/Bajar_el_ultimo**.

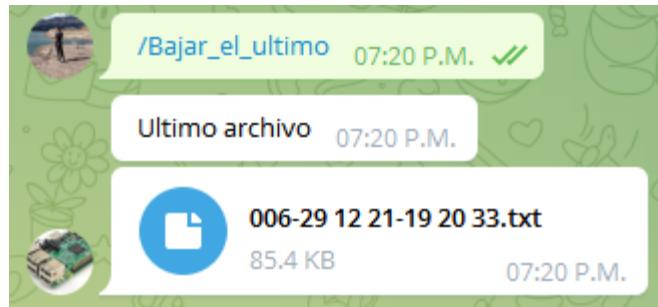


Figura 85: Descarga del último archivo generado.

Al cliquear sobre el archivo, automáticamente se descargará:

Wed Dec 29 19:20:33 2021									
Parámetros:									
Canal de disparo:	4								
Frecuencia de muestreo:	15000								
Nivel de disparo=	2.00								
Cantidad de muestras post trigger =	1000								
Cantidad de muestras pre trigger =	200								
Número de archivo =	6								
Volts/cuenta =	5000000/8388608								
Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) +	uS
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
4127430	4125379	4112659	4112270	4111199	4109368	4107854	4105699	1640816430	408650
4101519	4094235	4085622	4077752	4069688	4061174	4056468	4050241	1640816430	411958
4046861	4045583	4044252	4043050	4042059	4041589	4041432	4042123	1640816430	414796
4042970	4044032	4045153	4046119	4047512	4049512	4050889	4053656	1640816430	417629
4055898	4058294	4060508	4062704	4067919	4077205	4083847	4095390	1640816430	420475
4104369	4112057	4118274	4122735	4126488	4129620	4130815	4133204	1640816430	423312
4134517	4135326	4135240	4134233	4133051	4131521	4130107	4127887	1640816430	426151
4125617	4123324	4120656	4118285	4115533	4112679	4110513	4107619	1640816430	428986
4104532	4096735	4087989	4079854	4071303	4062027	4057186	4050394	1640816430	431830
4046229	4044700	4043271	4042113	4040970	4040239	4039978	4040595	1640816430	434668
4041464	4042512	4043739	4044729	4046093	4048180	4049508	4052482	1640816430	437507
4054859	4057446	4059844	4061989	4066315	4076004	4082862	4095024	1640816430	440363
4104649	4113331	4119926	4124907	4129214	4132879	4134238	4136862	1640816430	443182
4138399	4139429	4139588	4138648	4137390	4135694	4134203	4131773	1640816430	446019

- **Bajar los últimos 10 archivos generados:**

Para descargar los 10 archivos más nuevos generados, elija la opción número 2 a través del comando **/Bajar_ultimos_10**.

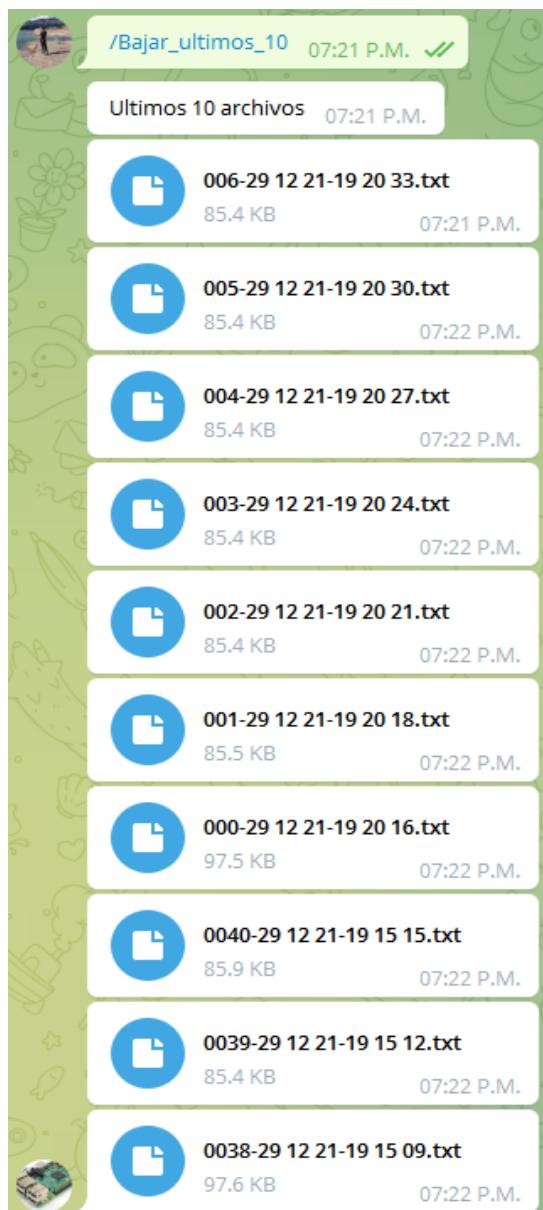


Figura 86: Descarga de los últimos 10 archivos generados.

Se descargarán en orden del archivo más nuevo al más antiguo. Puede observar que, aunque llegue al archivo número 0, de existir, seguirá por el mayor archivo numérico.

Si se ha accedido desde su celular, puede aparecer el siguiente mensaje:



En dicho caso elija la opción subir y podrá visualizar el archivo.

- **Listar los archivos:**

Si desea un listado de los archivos almacenados, elija la opción número 3 a través del comando **/Listar_archivos**. Primeramente, se presentará un listado de los últimos 10 archivos y las opciones de listar los archivos anteriores y posteriores.

Los archivos serán listados en orden del más antiguo al más nuevo.

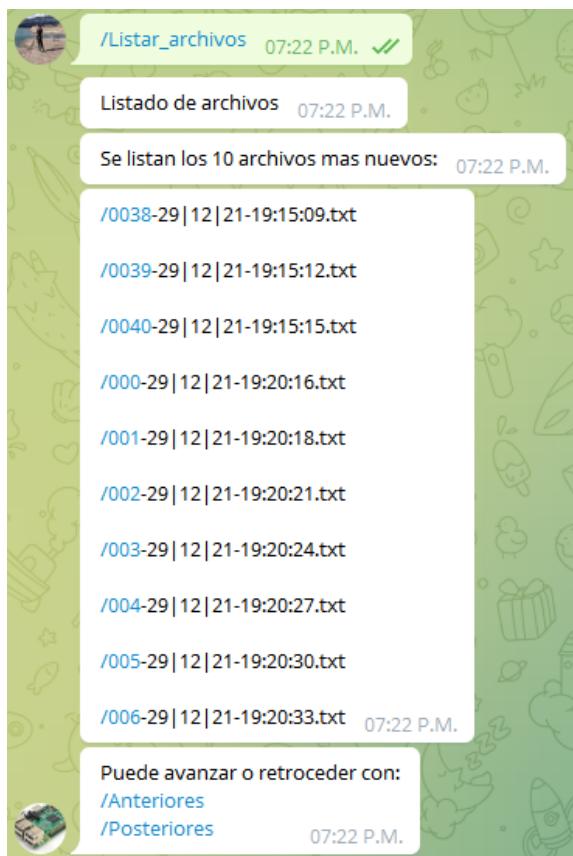


Figura 87: Listado de los últimos 10 archivos generados.

Si desea listar los 10 archivos numéricos anteriores, utilice el comando **/Anteriores**:

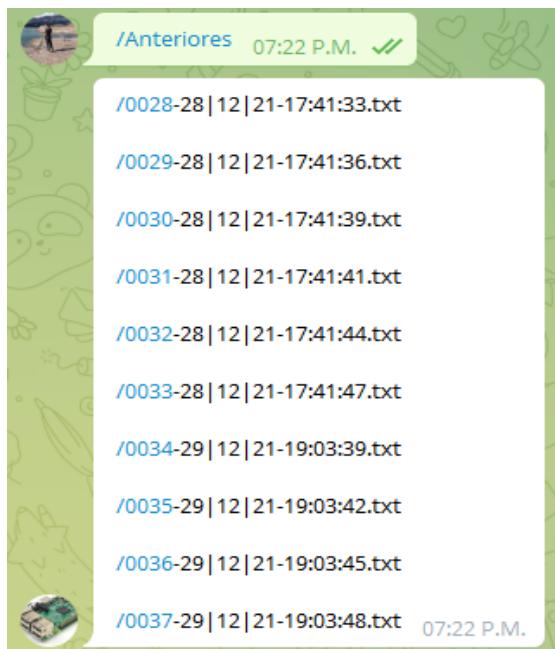


Figura 88: Listado de los 10 archivos anteriores.

Si desea listar los 10 archivos numéricos posteriores, utilice el comando **/Posteriores**:

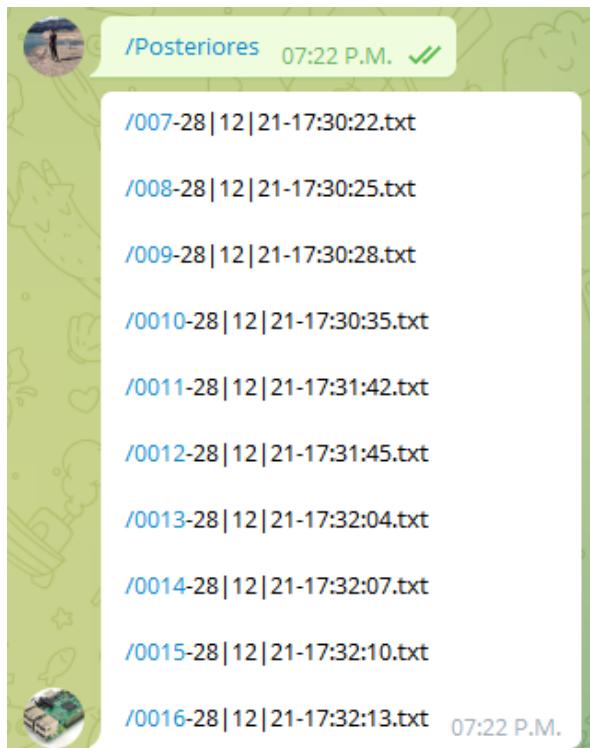


Figura 89: Listado de los 10 archivos posteriores.

Puede avanzar y retroceder las veces que desee para visualizar el listado completo de los archivos almacenados.

Cliqueando sobre los números de los archivos podrá descargar individualmente el archivo que deseé:

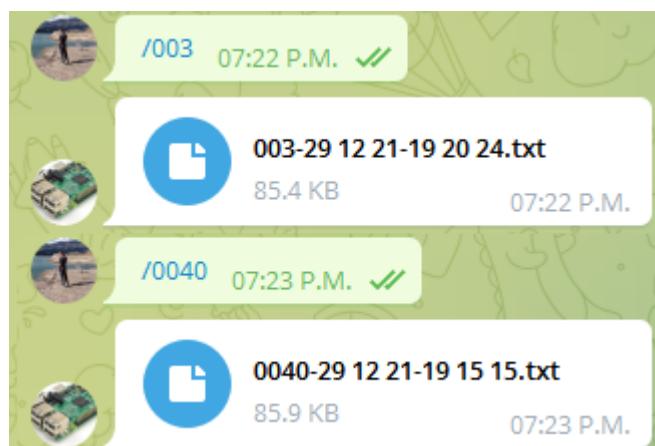


Figura 90: Descarga de un archivo particular del listado.

- **Opción B: Modificar parámetros**

Si desea modificar los parámetros de configuración elija la opción B a través del comando **/Parametros**. Se presentará el siguiente submenú:

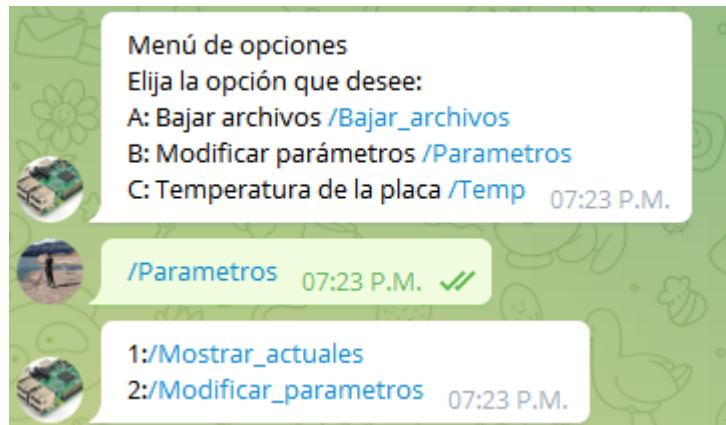


Figura 91: Menú de parámetros.

- **Mostrar los parámetros actuales:**

Si desea visualizar los parámetros actuales de configuración, elija la opción 1 a través del comando **/Mostrar_actuales**.

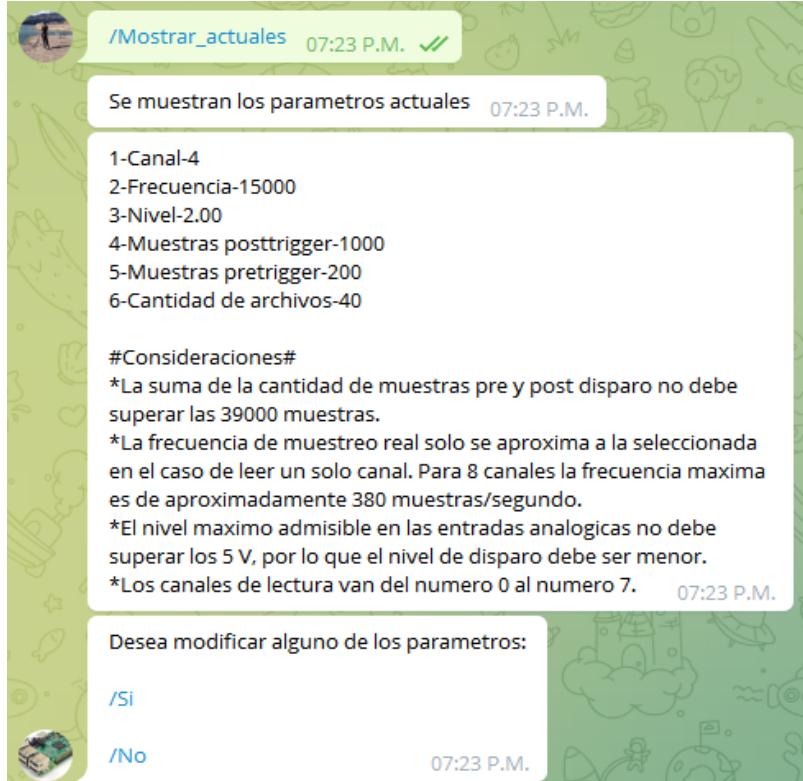


Figura 92: Visualización de parámetros actuales.

Se muestran además las consideraciones que deben tenerse en cuenta en la modificación de estos parámetros.

Adicionalmente se presenta la opción de modificar estos parámetros. Si envía el comando **/Si**, se le mostrará el menú con los parámetros a modificar:

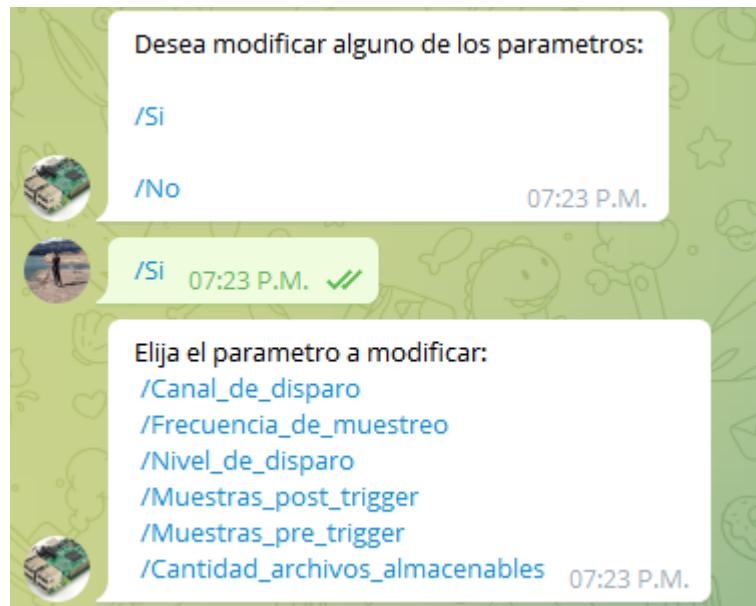


Figura 93: Modificación de parámetros.

Este menú es el mismo que se muestra en caso de elegir la opción 2 de modificar los parámetros, solo si el comando **/Si** es enviado luego de haber mostrado los parámetros actuales. Si envía el comando **/No**, no se mostrará ninguna nueva opción, pero cualquier comando posterior **/Si** no tendrá efecto en la modificación de estos parámetros.

- **Modificar los parámetros:**

Si lo que desea es modificar los parámetros de configuración elija la opción 2 a través del comando **/Modificar_parametros**:



- **Canal de disparo:** a través del comando **/Canal_de_disparo** puede seleccionar el canal que se monitoreará evaluando si alguna de sus mediciones supera el valor umbral de disparo y desencadena la generación de archivos.

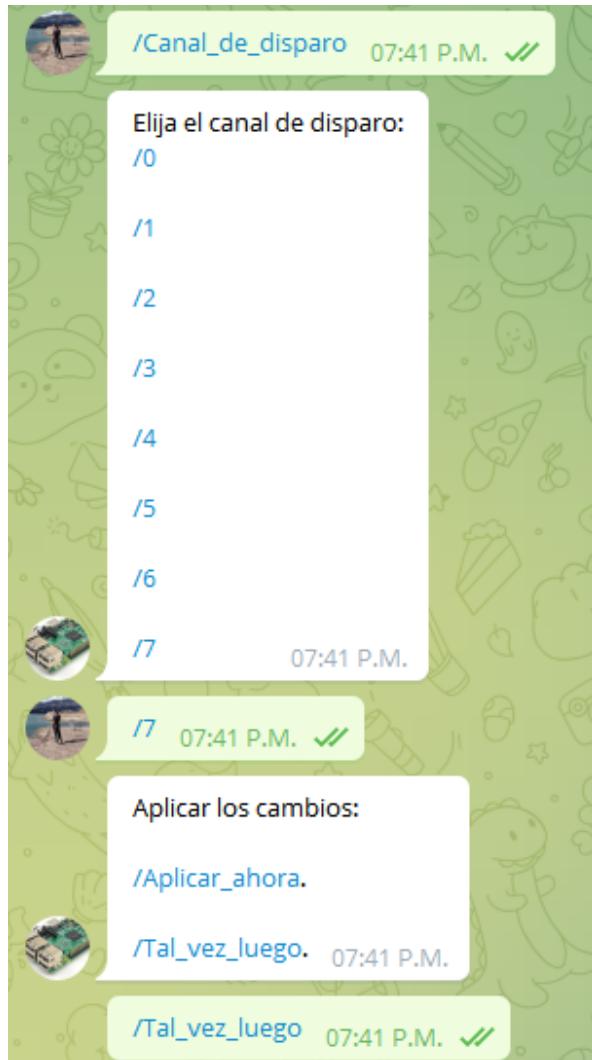


Figura 94: Elección del canal de disparo.

Se presentan los 8 canales del 0 al 7. Una vez seleccionado el canal, le aparecerá la opción de aplicar los cambios realizados o hacerlo luego. En caso de colocar el comando **/Aplicar_ahora** se envía automáticamente una señal al programa de medición para leer nuevamente los parámetros y ser aplicados donde correspondan. Si coloca el comando **/Tal_vez_luego**, simplemente se habrá modificado el archivo de configuración, pero el programa de medición seguirá utilizando los parámetros previos.

Estas opciones se muestran en cada parámetro.

- **Frecuencia de muestreo:** si desea modificar la frecuencia de muestreo, envíe el comando **/Frecuencia_de_muestreo**:

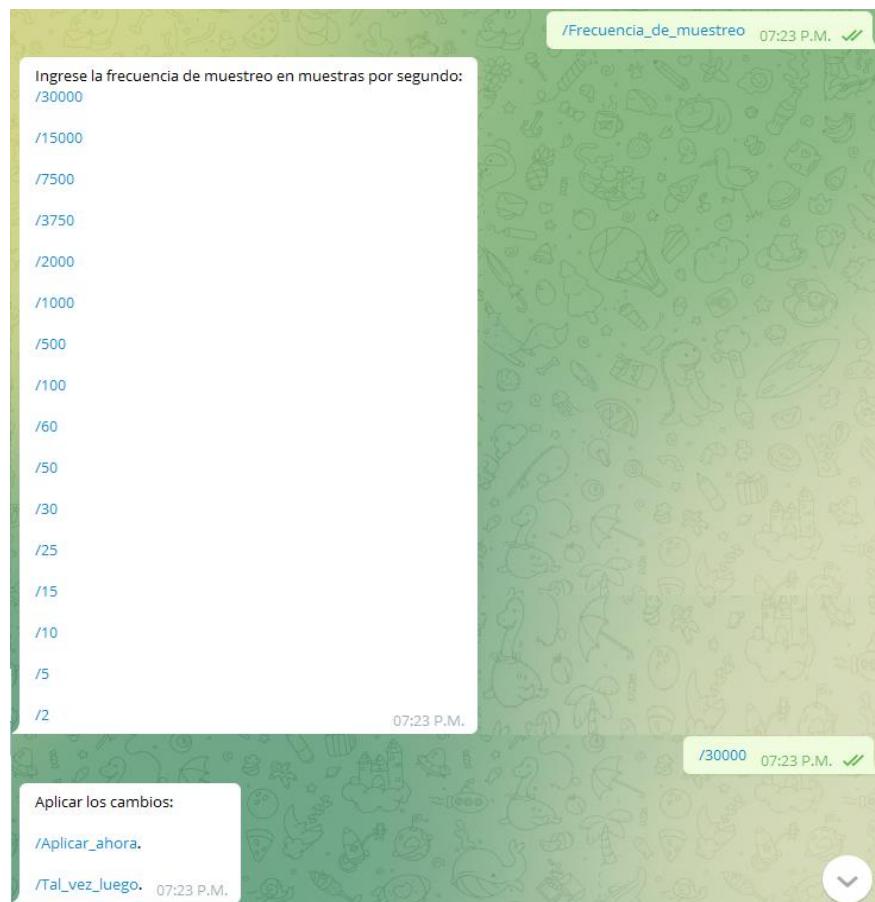


Figura 95: Elección de la frecuencia de muestreo.

Recordar que, aunque se muestren las frecuencias desde 2.5 SPS a 30000 SPS, la frecuencia de muestreo real es menor como se mencionó anteriormente.

Aunque se envíe el comando **/2** la frecuencia de muestreo real será de 2.5 SPS.

- **Nivel de disparo:** para modificar el nivel umbral de disparo envíe el comando **/Nivel_de_disparo:**

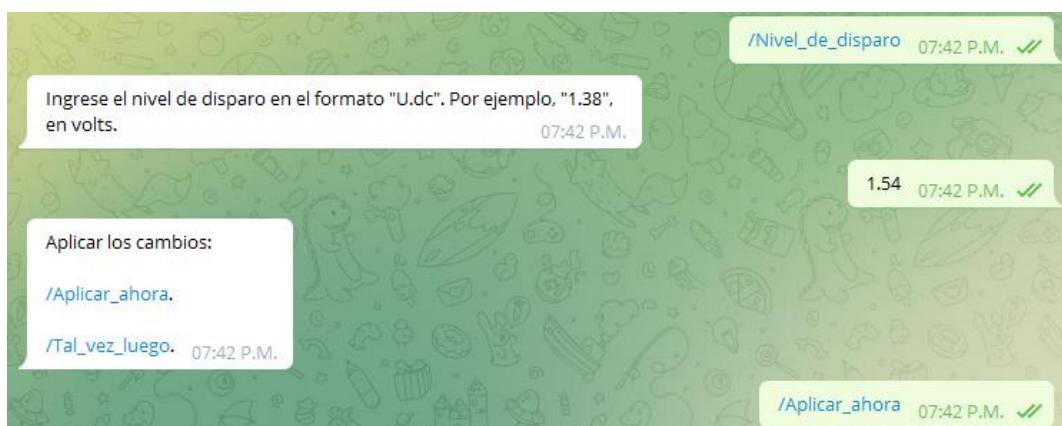


Figura 96: Elección del nivel de disparo.

El nivel debe ser ingresado en volts en el formato de una unidad y dos decimales.

- **Cantidad de muestras post disparo:** para modificar la cantidad de muestras luego del disparo envíe el comando **/Muestras_post_trigger**:



Figura 97: Elección de la cantidad de muestras post disparo.

Si la suma entre la cantidad de muestras post y pre disparo supera las 39000 muestras, se limitarán ambas cantidades a 18000 muestras.

- **Cantidad de muestras pre disparo:** para modificar la cantidad de muestras antes del disparo envíe el comando **/Muestras_pre_trigger**:

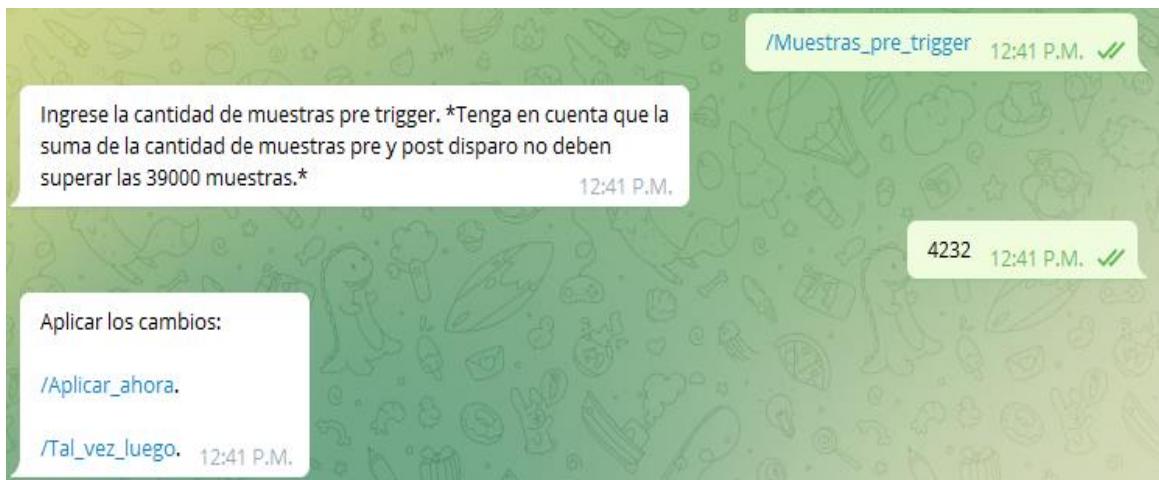


Figura 98: Elección de la cantidad de muestras pre disparo.

- **Cantidad de archivos almacenables:** para modificar la cantidad de archivos que se generarán antes de comenzar a sobrescribirlos, envíe el comando **/Cantidad_archivos_almacenables**:

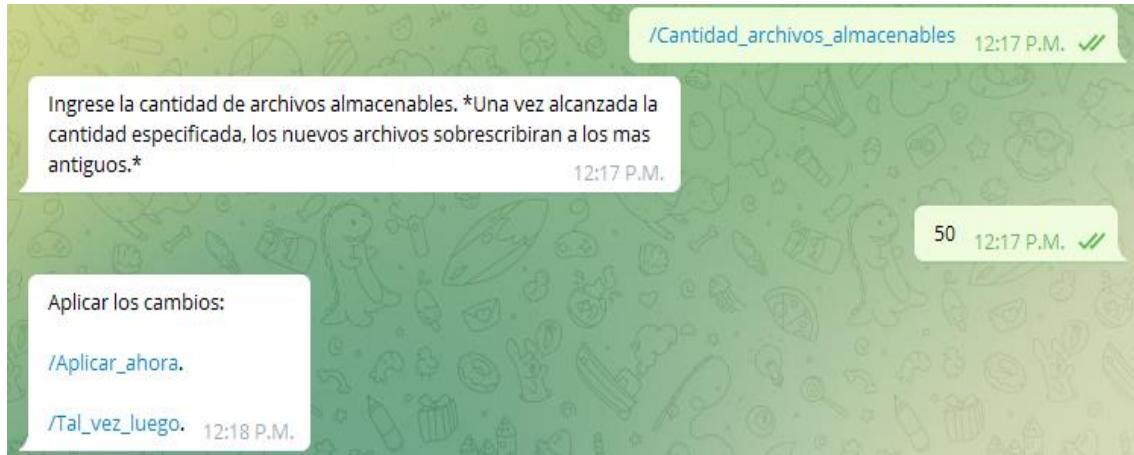


Figura 99: Elección de la cantidad de archivos almacenables.

- **Opción C: Visualizar la temperatura de la placa**

La tercera opción presentada en el menú principal es la de testear la temperatura interna de la placa Raspberry Pi. Para ello utilice el comando **/Temp**:



Figura 100: Visualización de la temperatura de la Raspberry Pi.

8 EJECUCIÓN AUTOMÁTICA

Ya se han desarrollado los dos programas principales; el programa de medición y el programa de comunicación remota vía Telegram. Debido a que el dispositivo está pensado para ser utilizado en estructuras en las que no siempre se encontrará presente el operario, a que sus periodos de uso serán prolongados y a que pueden existir interrupciones en la alimentación del dispositivo, es que los programas mencionados deben ejecutarse de forma automática en el inicio del dispositivo. De esta forma, si la placa Raspberry Pi sufre un reinicio y no se encuentra presente un operario para ejecutar estos programas, los mismos se ejecutarán de manera automática sin necesidad de hacerlo en forma manual.

Para ello estos programas deben ejecutarse como servicios o demonios.

Un **servicio** o **programa residente** o **daemon** es un tipo especial de programa que se ejecuta en segundo plano, en vez de ser controlado directamente por el usuario. Este tipo de programas continúa en el sistema, es decir, que puede ejecutarse de forma persistente o reiniciarse si se intenta matar el proceso dependiendo de su configuración.

Los demonios son útiles para hacer funcionar programas independientes de una sesión de usuario, procesos que se inicien de manera automática cuando el sistema arranca, servicios que permanecen a la escucha para ejecutar su tarea cuando son llamados.

Para realizar esto se utilizará el sistema **systemd** el cual es un sistema init y un administrador del sistema que se ha convertido en el nuevo estándar para las distribuciones Linux. La finalidad principal de un sistema init es inicializar los componentes que deben iniciarse tras arrancar el kernel Linux. El sistema init también se utiliza para administrar servicios y daemons para el servidor en cualquier momento mientras se ejecuta el sistema.

En systemd los demonios se definen y configuran en los llamados archivos de unidad, unit files. Estos archivos se alojan en `/etc/systemd/system/`. Para las tareas de administración de servicio, la unidad de destino serán unidades de servicio, que tienen archivos de unidad con un sufijo `.service`.

- **Creación y habilitación de un servicio:**

Para crear un nuevo archivo de servicio en systemd, escriba:

```
sudo nano /etc/systemd/system/name-of-your-service.service
```

Cada archivo systemd tiene tres secciones llamadas:

```
[Unit]  
[Service]  
[Install]
```

- **La sección de la Unidad:**

La sección Unidad proporciona información básica sobre el servicio y qué condiciones previas (por ejemplo, conectividad de red) deben cumplirse antes de que se inicie un servicio.

Description=Mi servicio

After=multi-user.target

En este caso, el servicio se ejecutará después de que las cuentas de usuario estén disponibles para el sistema durante el proceso de arranque.

O si desea que aguarde hasta que aparezca el escritorio.

After=graphical.target

Si el script requiere conectividad de red, utilice:

Requires=network.target

- La sección de servicio:

La sección Servicio proporciona instrucciones sobre cómo controlar el servicio.

Type=idle

La opción 'Tipo' de 'inactivo' le dice a systemd que espere hasta que todos los demás servicios se hayan completado. Esto espera hasta que el proceso de arranque se haya completado.

User=pi

El usuario define bajo qué usuario se ejecutará el programa.

ExecStart= sh /home/pi/Desktop/script.sh

Estos son los comandos y argumentos que se ejecutan cuando se inicia el servicio. Tenga en cuenta que debe proporcionar la ruta completa tanto del programa que está ejecutando como del script.

Reiniciar especifica lo que debería suceder en caso de que un servicio finalice intencionalmente o se bloquee.

Hay cuatro opciones de uso frecuente:

Restart=always # always restart

Restart=no # the service will not be restarted. This is the default.

Restart=on-success # Restart only when the service process exits cleanly (exit code 0)

```
Restart=on-failure # Restart only when the service process does not exit cleanly  
(node-zero exit code)
```

Esto es seguido por:

```
RestartSec=60
```

;que especifica el tiempo en segundos a esperar antes de reiniciar un servicio.

- La sección de instalación:

La sección de instalación proporciona instrucciones sobre cómo systemctl activa el servicio personalizado si está habilitado con systemctl enable. Esto se usa principalmente para iniciar el servicio personalizado en el arranque.

Una sección de instalación típica resultante se ve así:

```
[Install]  
WantedBy=multi-user.target
```

Se creó un servicio que, al arranque del sistema, ejecuta en forma directa los ejecutables del programa de medición y el de comunicación remota.

- **Servicio de medición:**

Para el programa de medición se generó el siguiente servicio llamado **raspy.service**:

```
[Unit]  
Description= Servicio de Medición  
After= multi-user.target  
  
[Service]  
Type=simple  
Restart=always  
RestartSec=3  
ExecStart= sudo ./home/pi/Desktop/Archivos/libreria/RaspberryPi-  
ADC-DAC/build/testAdda  
#ExecReload=/bin/kill -HUP $MAINPID  
  
[Install]  
WantedBy=multi-user.target
```

Este servicio se ejecutará luego de que estén disponibles las cuentas de usuario. Lo ejecutará el usuario pi y se reiniciará siempre que sufra un error o interrupción luego de esperar 3 segundos.

- **Servicio de Telegram:**

Para el programa de comunicación remota con Telegram se creó el servicio **telegram.service**:

```
[Unit]
Description= Servicio Telegram
Requires=network.target
After=network.target

[Service]
Type=simple
Restart=always
RestartSec=3
ExecStart= sudo ./home/pi/Desktop/Archivos/telegrambot/telebot-
master/test/test/echobot
#ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

Posee las mismas características que el servicio anterior, pero se iniciará luego de que haya conexión a internet.

- **Inicio del servicio:**

Para la creación del servicio ejecute el comando ya explicado:

```
sudo nano /etc/systemd/system/raspy.service
```

Configurar los permisos del archivo:

```
sudo chmod 755 /etc/systemd/system/raspy.service
```

Cargar el servicio al sistema init:

```
sudo systemctl daemon-reload
```

Iniciar el servicio:

```
sudo systemctl start raspy.service
```

Habilitar el servicio para que se ejecute automáticamente en el arranque del sistema:

```
sudo systemctl enable raspy.service
```

Con estos pasos el servicio ya se encuentra en funcionamiento. Si desea ver el estado del servicio ejecute:

```
sudo systemctl status raspy.service
```

```

● raspy.service - Servicio Medición
   Loaded: loaded (/etc/systemd/system/raspy.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-02-04 11:41:20 -03; 40s ago
     Main PID: 789 (sh)
        Tasks: 3 (limit: 2062)
       CGroup: /system.slice/raspy.service
           └─789 /usr/bin/sh /home/pi/Desktop/script1.sh
             ├─794 sudo ./testAdda
             └─796 ./testAdda

Feb 04 11:41:20 raspberrypi systemd[1]: Started Servicio Medición.
Feb 04 11:41:20 raspberrypi sudo[794]:          pi : TTY=unknown ; PWD=/home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build ; USER=root ; COMMAND=./testAdda
Feb 04 11:41:20 raspberrypi sudo[794]: pam_unix(sudo:session): session opened for user root by (uid=0)
~
~
~
```

Figura 101: Estado de ejecución del servicio de medición.

Donde puede observarse que se encuentra en estado activo y ejecutándose. Lo mismo para el servicio de Telegram:

```

● telegram.service - Servicio Telegram
   Loaded: loaded (/etc/systemd/system/telegram.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-02-04 11:47:49 -03; 25s ago
     Main PID: 1580 (sh)
        Tasks: 3 (limit: 2062)
       CGroup: /system.slice/telegram.service
           └─1580 /usr/bin/sh /home/pi/Desktop/script2.sh
             ├─1581 sudo ./echobot
             └─1582 ./echobot

Feb 04 11:47:49 raspberrypi systemd[1]: Started Servicio Telegram.
Feb 04 11:47:49 raspberrypi sudo[1581]:          root : TTY=unknown ; PWD=/home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test ; USER=root ; COMMAND=./echobot
Feb 04 11:47:49 raspberrypi sudo[1581]: pam_unix(sudo:session): session opened for user root by (uid=0)
~
~
```

Figura 102: Estado de ejecución del servicio de Telegram.

Si desea detener el servicio:

```
sudo systemctl stop raspy.service
```

O reiniciarlo:

```
sudo systemctl restart raspy.service
```

Los mismos comandos fueron ejecutados para el servicio de Telegram.

Importante: si desea ejecutar los programas desde la terminal de comandos en forma manual, primero debe detener estos servicios. Caso contrario, el programa de medición fallará debido a que ya existe otro proceso utilizando el ADC y el programa de comunicación remota enviará los mismos mensajes y descargará los mismos archivos dos veces debido a que se ejecuta simultáneamente dos veces.

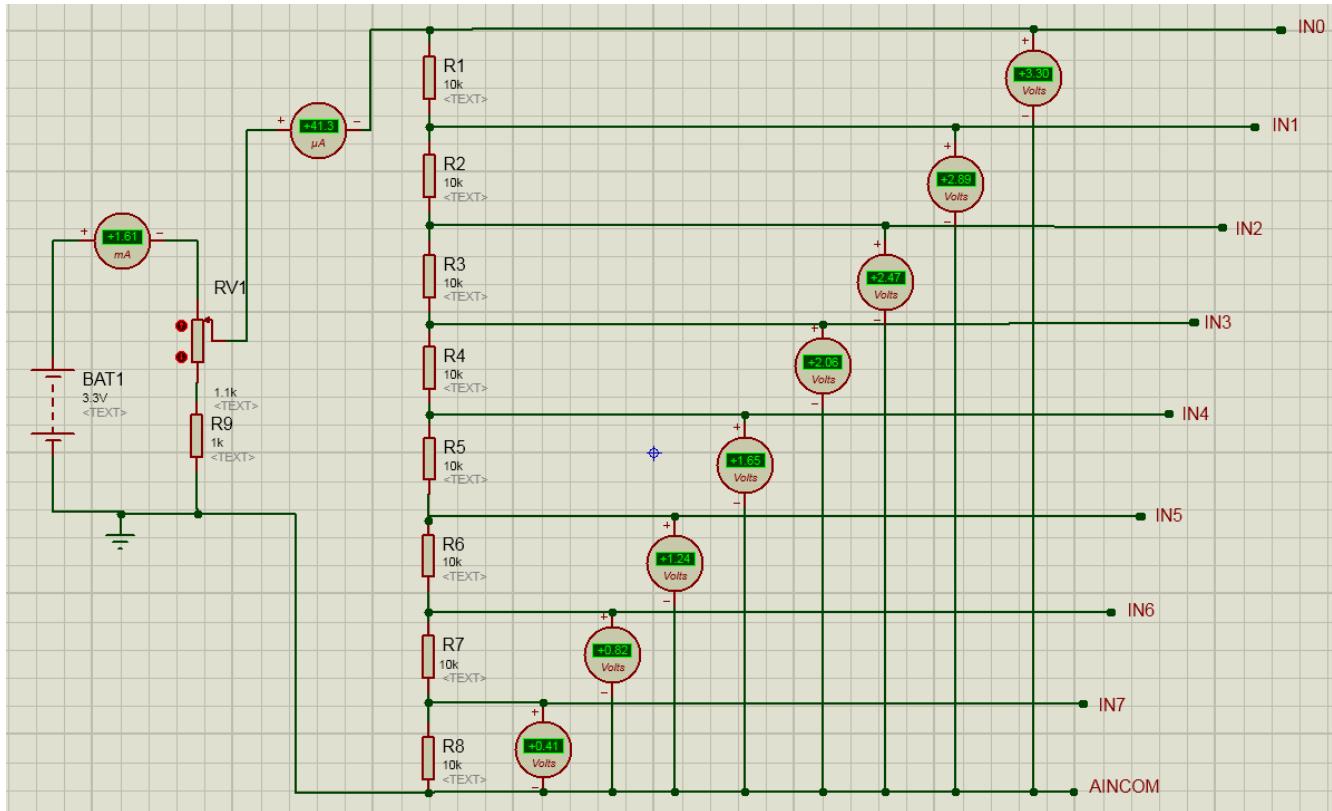
9 PRUEBAS REALIZADAS

Se realizaron una gran cantidad de pruebas con el fin de determinar las especificaciones reales del dispositivo. Para ello se lo sometió a diversas situaciones y escenarios. Las pruebas más relevantes se muestran a continuación.

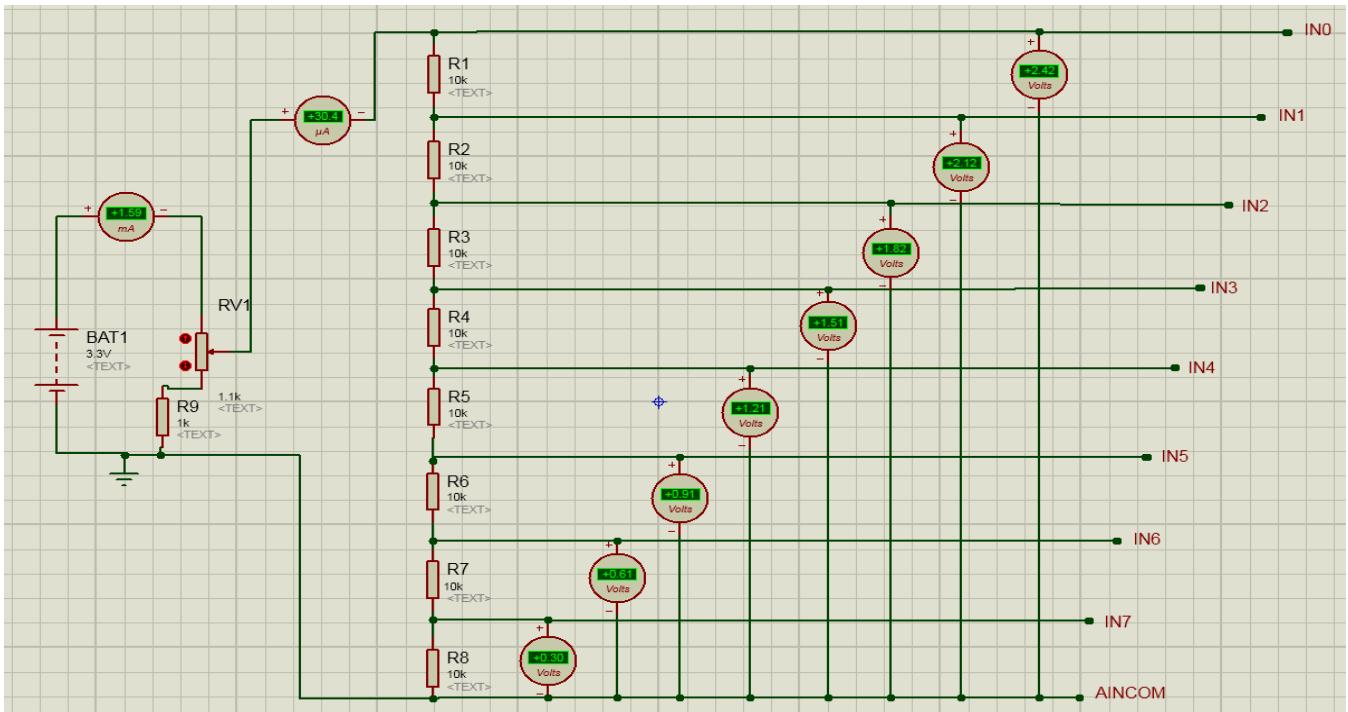
▪ Comparación con el simulador Proteus

Se propuso el siguiente circuito para medir dentro de todo el rango posible:

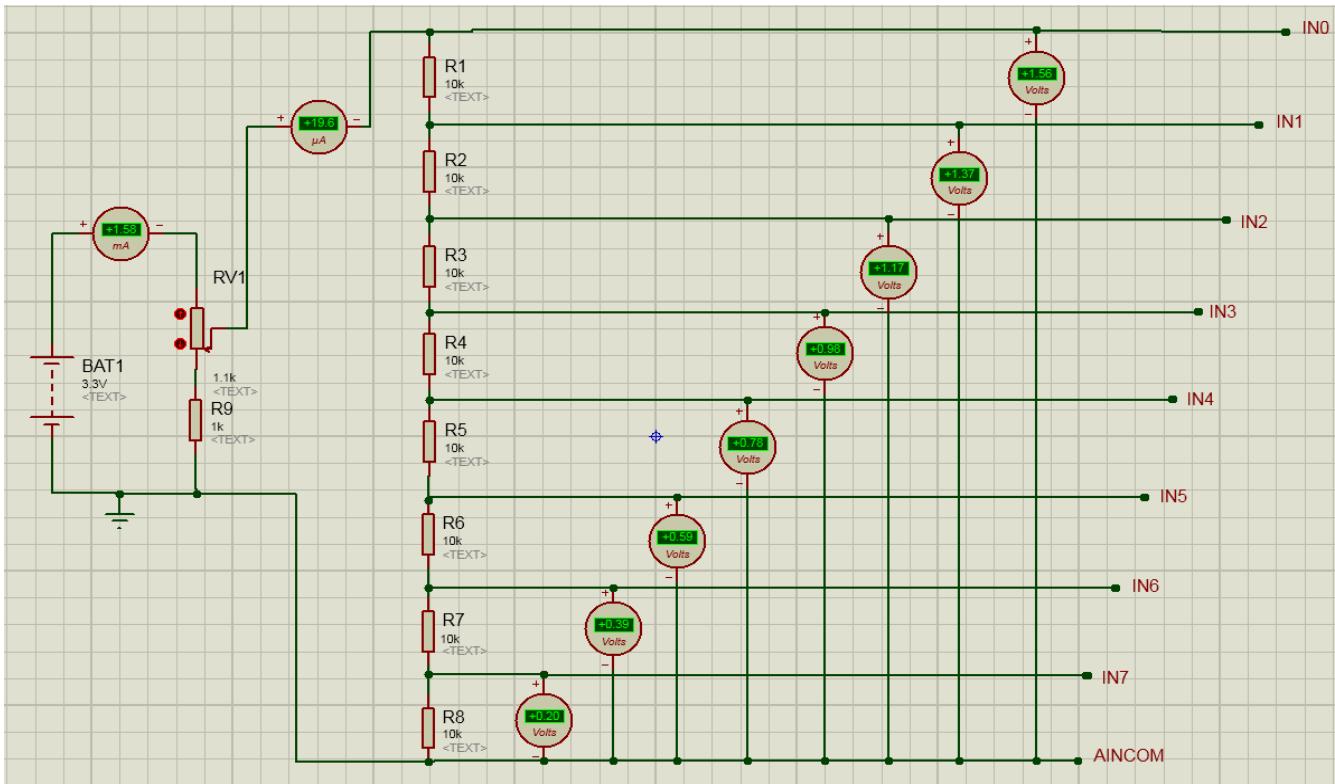
Posición 1:



Posición 2:



Posición 3:



Posición 4:

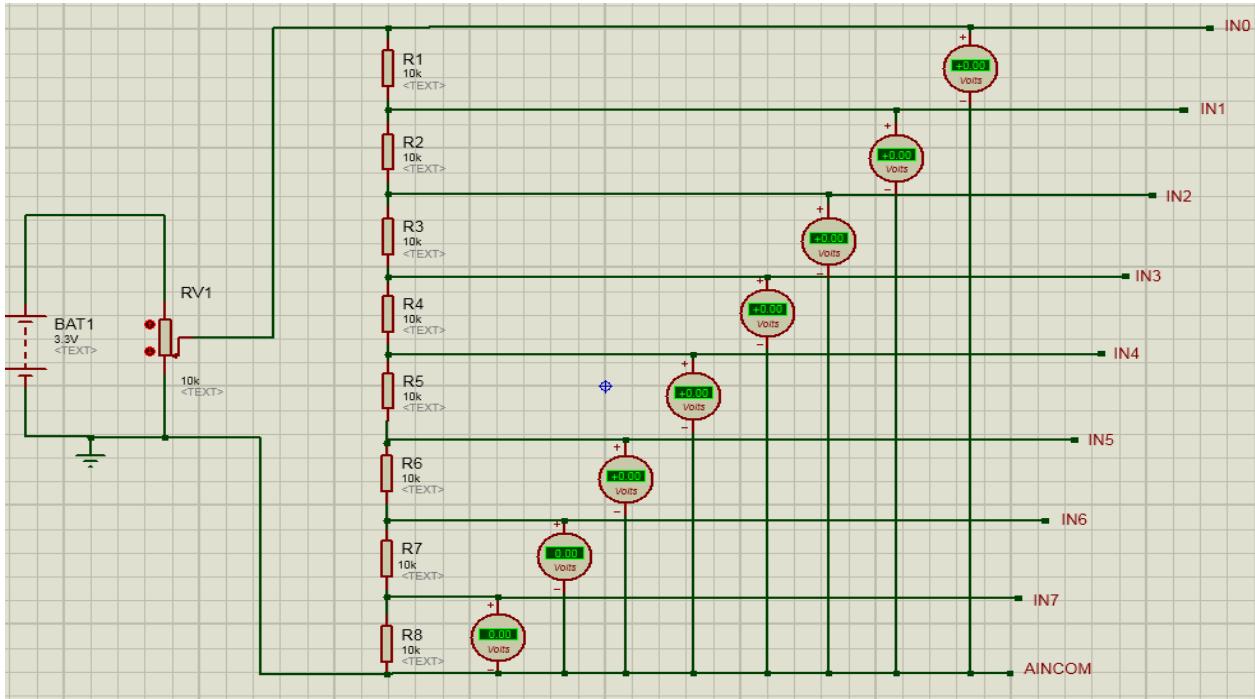


Figura 103: Simulación de circuito de prueba en Proteus.

Con este arreglo se logró probar todas las entradas en forma simultánea, sin exceder la tensión máxima y, además, al regular el potenciómetro se obtuvieron distintos valores. El último caso también permitió verificar qué tan cercano a los 0 V puede medir el dispositivo.

El primer circuito realizado con la Raspberry Pi y la placa conversora es:

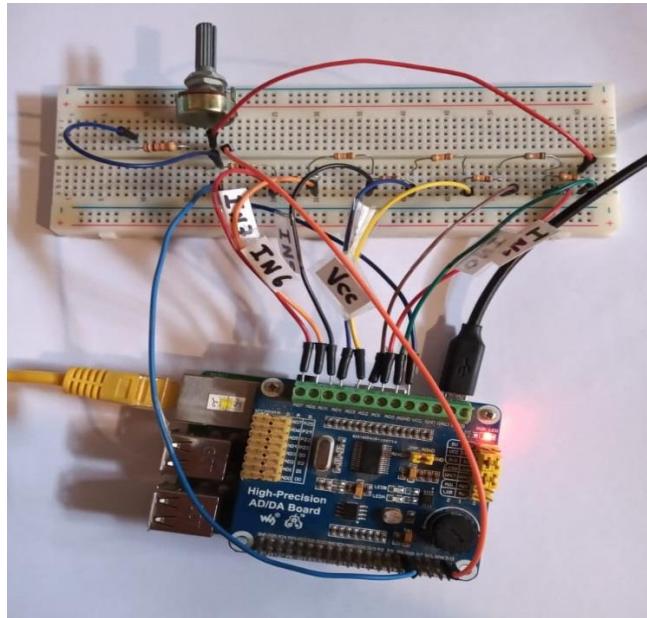


Figura 104: Circuito preliminar de prueba.

Los resultados preliminares obtenidos fueron:

- **30000 muestras/seg**

Gain 1:

Posición 1:

0 : 3.305140 V 1 : 2.876480 V 2 : 2.461770 V 3 : 2.051114 V 4 : 1.641852 V 5 :
1.235892 V 6 : 0.829658 V 7 : 0.425504 V

Posición 2:

0 : 2.434124 V 1 : 2.119337 V 2 : 1.816245 V 3 : 1.515510 V 4 : 1.215767 V 5 :
0.918506 V 6 : 0.621250 V 7 : 0.326598 V

Posición 3:

0 : 1.540607 V 1 : 1.348282 V 2 : 1.159013 V 3 : 0.969002 V 4 : 0.778720 V 5 :
0.589560 V 6 : 0.400530 V 7 : 0.213876 V

Posición 4:

0 : 0.019494 V 1 : 0.028522 V 2 : 0.033787 V 3 : 0.035152 V 4 : 0.034402 V 5 :
0.032755 V 6 : 0.030974 V 7 : 0.031661 V

Gain 2:

Posición 1:

0 : 4.999999 V 1 : 4.999999 V 2 : 4.831027 V 3 : 4.033009 V 4 : 3.240193 V 5 :
2.454373 V 6 : 1.667390 V 7 : 0.887452 V

Posición 2:

0 : 4.839152 V 1 : 4.197491 V 2 : 3.591684 V 3 : 3.000895 V 4 : 2.416704 V 5 :
1.838211 V 6 : 1.259764 V 7 : 0.689611 V

Posición 3:

0 : 3.071092 V 1 : 2.686755 V 2 : 2.315361 V 3 : 1.949405 V 4 : 1.580867 V 5 :
1.215804 V 6 : 0.851453 V 7 : 0.516521 V

Posición 4:

0 : 0.058841 V 1 : 0.095810 V 2 : 0.118761 V 3 : 0.125831 V 4 : 0.125176 V 5
: 0.121325 V 6 : 0.116766 V 7 : 0.121846 V

Se observa que las mediciones tomadas con el dispositivo se encuentran muy cercanas a las simuladas dentro de todo el rango de medición. Sin embargo, cuando el nivel es cercano a los 0 V o inclusive, las mediciones no son realmente de potencial nulo. Esto es debido a la alta impedancia de las entradas analógicas del conversor, las cuales captan pequeñas interferencias y acusan niveles significativos de señal. La mejor forma de disminuir este error es disminuir la incidencia de estas interferencias entrantes a través de los métodos expuestos posteriormente (véase sección ‘Contrastación con multímetro de banco’).

▪ Cálculo de la desviación estándar

Se debe calcular la desviación estándar de las muestras respecto a la media en cada uno de los canales.

Para ello se utilizará la siguiente fórmula:

$$DE_{muestra} = \sqrt{\frac{\sum |x - \bar{x}|^2}{n - 1}}$$

Donde:

- X es cada una de las muestras tomadas en cada canal;
- \bar{x} es la media de las muestras adquiridas;
- n es la cantidad de muestras tomadas.

Hay dos formas de realizar el cálculo a través de un programa desarrollado para tal fin. La primera consiste en almacenar todas las muestras y realizar los cálculos al final del bucle. Esta primera forma exige menos procesamiento ya que el cálculo se realiza al final una sola vez, pero también requiere más memoria pues se deberá almacenar todas las muestras en el tiempo, para cada canal.

La segunda consiste en calcular la media, la desviación de cada muestra y el cuadrado de la distancia a la media para cada dato en cada iteración del bucle. Este método requiere mayor procesamiento, pero menos memoria.

Para demostrar ambos métodos recurriremos a un ejemplo. Supongamos que poseemos 6 muestras: 5, 4, 6, 8, 1, 3

Método 1:

Primero calculamos la media de las muestras:

$$\bar{x} = \frac{5 + 4 + 6 + 8 + 1 + 3}{6} = 4.5$$

Luego calculamos la suma de los cuadrados de las distancias de cada muestra a la media:

$$\sum_{i=0}^{n-1} (x_i - \bar{x})^2 = (5 - 4.5)^2 + (4 - 4.5)^2 + (6 - 4.5)^2 + (8 - 4.5)^2 + (1 - 4.5)^2 + (3 - 4.5)^2 = 29.5$$

Y, por último, calculamos la desviación estándar:

$$DesvEst = \sqrt{\frac{29.5}{6 - 1}} = 2.42899156$$

Método 2:

Definimos:

- $\Delta = x - \bar{x}$
- $\bar{x} = \bar{x}_{\text{anterior}} + \Delta/n$
- $m2 = m2_{\text{anterior}} + \Delta * (x - \bar{x})$

x	n	Δ	\bar{x}	$m2$
5	1	5-0=5	0+5/1=5	0+5*(5-5) =0
4	2	4-5=-1	5-1/2=4.5	0+(-1) *(4-4.5) =0.5
6	3	6-4.5=1.5	4.5+1.5/3=5	0.5+1.5*(6-5) =2
8	4	8-5=3	5+3/4 = 5.75	2+3*(8-5.75) = 8.75
1	5	1-5.75=-4.75	5.75-4.75/5=4.8	8.75+(-4.75) *(1-4.8) =26.8
3	6	3-4.8=-1.8	4.8+(-1.8) /6=4.5	26.8+(-1.8)*(3-4.5) =29.5

Estos son los cálculos realizados en cada iteración del bucle for. Con los últimos datos obtenidos:

$$DesvEst = \sqrt{\frac{29.5}{6 - 1}} = 2.42899156$$

Y se obtiene exactamente el mismo resultado.

El código para hacer estos cálculos en todos los canales es:

```
n++;

////Calculo de la media, el delta y la distancia al cuadrado de
cada canal////
for (Loop = 0; Loop <NChannels; Loop++)
{      delta[Loop]=volt[Loop]-media[Loop];      }

for (Loop = 0; Loop <NChannels; Loop++)
{      media[Loop]+=delta[Loop]/n;      }

for (Loop = 0; Loop <NChannels; Loop++)
{m2[Loop]+=(delta[Loop]*(volt[Loop]-media[Loop])); }

for (Loop = 0; Loop <NChannels; Loop++)
{      varianza[Loop]=m2[Loop]/((double)n-1); }

for (Loop = 0; Loop <NChannels; Loop++)
{      stdev[Loop]= sqrt(varianza[Loop]);      }
```

En el cálculo de la desviación estándar aparecieron problemas de compilación con la función `sqrt()` que realiza la raíz cuadrada de la varianza.

```
pi@raspberrypi:~/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build $ make
[ 42%] Built target ADDA_WS_RPI
[ 57%] Linking C executable testAdda
/usr/bin/ld: CMakeFiles/testAdda.dir/Prueba-Fer.c.o: in function `main':
Prueba-Fer.c:(.text+0x494): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```

La solución a esto fue agregar al final del comando de compilación el indicador del enlazador `-lm`, que incluye en dicha compilación todas las librerías especificadas como `math.h` que es donde se encuentra esta función.

Un enlazador es un programa que toma los objetos generados en los primeros pasos del proceso de compilación, la información de todos los recursos necesarios (biblioteca), quita aquellos recursos que no necesita, y enlaza el código objeto con su(s) biblioteca(s) con lo que finalmente produce un fichero ejecutable o una biblioteca. En el caso de los programas enlazados dinámicamente, el enlace entre el programa ejecutable y las bibliotecas se realiza en tiempo de carga o ejecución del programa.

Este indicador de enlazador no podemos colocarlo al final de make. Debemos modificar uno de los archivos que utiliza el compilador para realizar la compilación.

Para ello se debe ir a la siguiente ruta:

/home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC-master/build/CMakeFiles/testAdda.dir y modificamos el archivo link.txt.

```
estAdda.dir/Prueba-Fer.c.o -o testAdda AD-DA-WS-RPI/libADDA_WS_RPI.a /usr/local/lib/libbcm2835.a -lm
```

Y el error fue solucionado:

```
pi@raspberrypi:~/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build $ make
[ 42%] Built target ADDA_WS_RPI
[ 57%] Linking C executable testAdda
/usr/bin/ld: CMakeFiles/testAdda.dir/Prueba-Fer.c.o: in function `main':
Prueba-Fer.c:(.text+0x494): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
make[2]: *** [CMakeFiles/testAdda.dir/build.make:86: testAdda] Error 1
make[1]: *** [CMakeFiles/Makefile2:98: CMakeFiles/testAdda.dir/all] Error 2
make: *** [Makefile:84: all] Error 2
pi@raspberrypi:~/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build $ make
[ 42%] Built target ADDA_WS_RPI
[ 57%] Linking C executable testAdda
[ 71%] Built target testAdda
```

Al ejecutar e imprimir los valores de m2 y la varianza del primer canal, junto con las desviaciones estandar de todos los canales obtenemos:

```
init done !
m2= 0.136118
var= 0.068059
ADC_DAC_Close
Desviaci n estandar:
    Canal 0: 0.260881
    Canal 1: 0.386856
    Canal 2: 0.183271
    Canal 3: 0.705527
    Canal 4: 0.485077
    Canal 5: 0.551943
    Canal 6: 0.365837
    Canal 7: 0.066234
Test ADDA finished with returned code 0
```

Figura 105: C culo inicial de las desviaciones est ndar de cada canal.

Por lo tanto, las desviaciones estándar mostradas para cada canal son:

- Canal 0: 0.260881
- Canal 1: 0.386856
- Canal 2: 0.183271
- Canal 3: 0.705527
- Canal 4: 0.485077
- Canal 5: 0.551943
- Canal 6: 0.365837
- Canal 7: 0.066234

Es importante aclarar que si se trabaja con las mediciones en volts, las desviaciones de las mediciones serán tan pequeñas que se aproximarán a cero. Para evitar esta falta de precisión se deberá trabajar en milivolts o microvolts.

Las desviaciones estándar en volts antes calculadas se realizaron en base a una población de 20 muestras. Debido a que este conjunto es muy pequeño en comparación a la cantidad de muestras que se obtendrán normalmente, será mejor calcular la desviación estándar sobre una población mucho mayor de muestras y realizar los cálculos en Excel.

▪ **Prueba de los 100 segundos**

Anteriormente se mencionó que la cantidad máxima de muestras que podían almacenarse en un arreglo sin llenar la memoria RAM era de aproximadamente 40000 muestras, lo que, a la frecuencia máxima equivalía a poco más de 102 segundos de medición constante.

Para el cálculo de la desviación estándar sobre una población mucho mayor a la tomada anteriormente, se tomaron 39000 muestras, lo que equivale aproximadamente a 100 segundos de medición.

Los resultados obtenidos fueron los siguientes:

Tiempo (S)	Canal 0	Canal 1	Canal 2	Canal 3	Canal 4	Canal 5	Canal 6	Canal 7	Tiempo: canal 0 (S) +	uS	Delta t	frecuencia
0,000000	666987	1283196	1893110	2496051	3103386	3709783	4327294	5000295,00	1626445133	718814	0,003124	320,102433
0,003124	661874	1275588	1881915	2483529	3093652	3704794	4324077	4999890,00	1626445133	721938	0,003152	317,258883
0,006276	658905	1268824	1874749	2476402	3085411	3699900	4321450	4999948,00	1626445133	725090	0,003094	323,206206
0,009370	656938	1265312	1870443	2471887	3081678	3694085	4315410	5000085,00	1626445133	728184	0,003187	313,77471
0,012557	655325	1264221	1868397	2469479	3079755	3692597	4313752	5000185,00	1626445133	731371	0,003309	302,206105
0,015866	654624	1261454	1865827	2467302	3077289	3693569	4318378	4999928,00	1626445133	734680	0,003103	322,268772
0,018969	654313	1260733	1864974	2466389	3076770	3693519	4318676	4999747,00	1626445133	737783	0,003148	317,662008
0,022117	654353	1262134	1865333	2466818	3077311	3694420	4319053	5000132,00	1626445133	740931	0,003318	301,386377
0,025435	652688	1259687	1864335	2466655	3077783	3694587	4318720	4999908,00	1626445133	744249	0,003103	322,268772
0,028538	654170	1260685	1865002	2466705	3077102	3693923	4318647	4999973,00	1626445133	747352	0,003091	323,519896
0,031629	1191578	1264841	1865831	2467406	3077663	3694538	4323465	5000248,00	1626445133	750443	0,003232	309,405941
0,034861	657048	1263306	1866935	2468082	3078651	3695288	4314874	5000114,00	1626445133	753675	0,003255	307,219662
0,038116	654626	1261248	1865552	2467052	3077397	3694028	4318428	4999893,00	1626445133	756930	0,003109	321,646832
0,041225	654371	1863026	1867759	2468965	3079663	3696165	4319726	5000779,00	1626445133	760039	0,003243	308,35646
0,044468	647481	1255400	1862296	2466668	3079362	3699188	4328609	50001017,00	1626445133	763282	0,002663	375,516335
0,047131	647999	1255578	1861670	2465696	3078511	3699032	4328420	50001018,00	1626445133	765945	0,002693	371,333086
0,049824	648187	1255598	1861800	2465730	3078715	3699316	4329593	5001263,00	1626445133	768638	0,002626	380,807312
0,052450	648310	1255882	1861879	2465883	3078757	3699452	4329435	5001474,00	1626445133	771264	0,002580	387,596899
0,055030	648530	1255886	1862124	2465959	3078766	3699232	4329408	5001351,00	1626445133	773844	0,002590	386,100386
0,057620	648430	1255994	1862110	2465822	3078703	3699043	4329501	5001322,00	1626445133	776434	0,002581	387,446726
0,060201	648380	1255935	1862204	2465946	3078851	3698924	4328291	5000761,00	1626445133	779015	0,002772	360,750361
0,062973	648275	1255715	1861868	2465984	3079000	3699277	4329409	5001152,00	1626445133	781787	0,002640	378,787879
0,065613	648638	1256113	1862263	2466083	3079162	3699517	4329304	5001212,00	1626445133	784427	0,002646	377,928949
0,068259	648573	1256247	1862456	2466243	3079280	3699685	4329664	5001329,00	1626445133	787073	0,002599	384,763371
0,070858	648506	1256112	1862283	2466133	3079123	3699356	4329640	5001334,00	1626445133	789672	0,002581	387,446726
0,073439	648527	1255912	1862123	2465953	3078855	3699486	4329433	5001398,00	1626445133	792253	0,002571	388,953715
0,076010	648482	1255957	1862111	2465822	3078704	3699223	4329583	5001325,00	1626445133	794824	0,002573	388,65138

Desviaciones estándar:

	Media canal 7	Desviación est. canal 7
Cuentas	5014367.83	2083.964334
Volts	2.988796	0.00124214
Porcentaje		0.04%

	Media canal 6	Desviación est. canal 6
cuentas	4342568.40	1497.264515
volts	2.588372	0.000892439
porcentaje		0.03%

	Media canal 5	Desviación est. canal 5
cuentas	3709739.59	4415.362001
volts	2.211177	0.002631761
porcentaje		0.12%

	Media canal 4	Desviación est. canal 4
cuentas	3085619.06	3361.108064
volts	1.839172	0.002003377
porcentaje		0.11%

	Media canal 3	Desviación est. canal 3
cuentas	2469698.20	2579.756678
volts	1.472055	0.001537655
porcentaje		0.10%

	Media canal 2	Desviación est. canal 2
cuentas	1863056.46	2174.971895
volts	1.110468	0.001296384
porcentaje		0.12%

	Media canal 1	Desviación est. canal 1
cuentas	1254268.97	451.2136127
volts	0.747603	0.000268944
porcentaje		0.04%

	Media canal 0	Desviación est. canal 0
cuentas	644480.15	421.1913972
volts	0.384140	0.00025105
porcentaje		0.07%

Se han calculado las desviaciones estándar para cada canal en cuentas (resultado arrojado en forma directa por el conversor), en volts y porcentualmente. Puede observarse que debido a que la población tomada es mucho mayor que en el caso anterior, la desviación estándar en volts es significativamente menor y muy aceptable para los estándares del dispositivo.

_Gráficos:

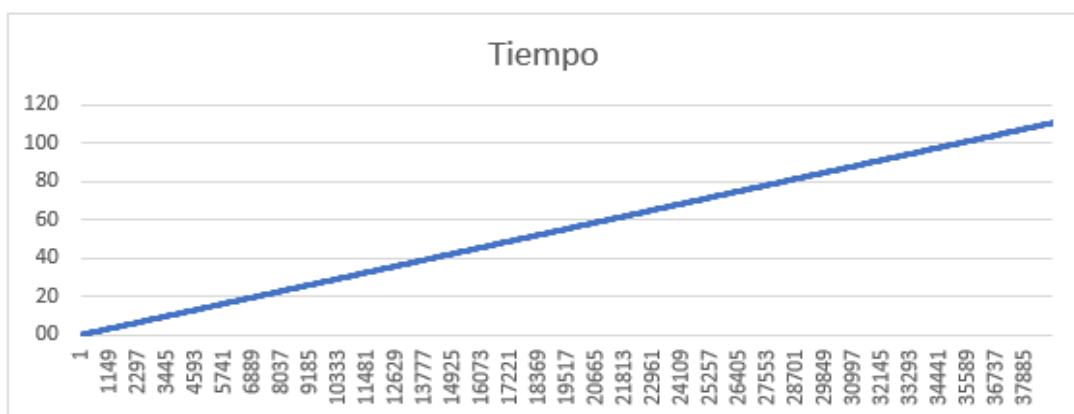


Figura 106: Variación temporal de las muestras.

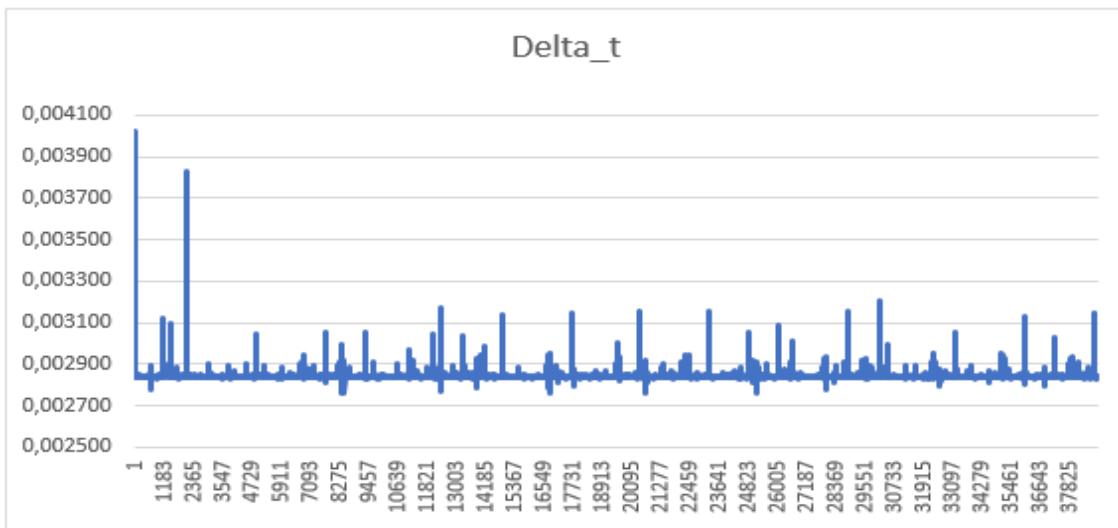


Figura 107: Variación del periodo de muestreo en función de las muestras.

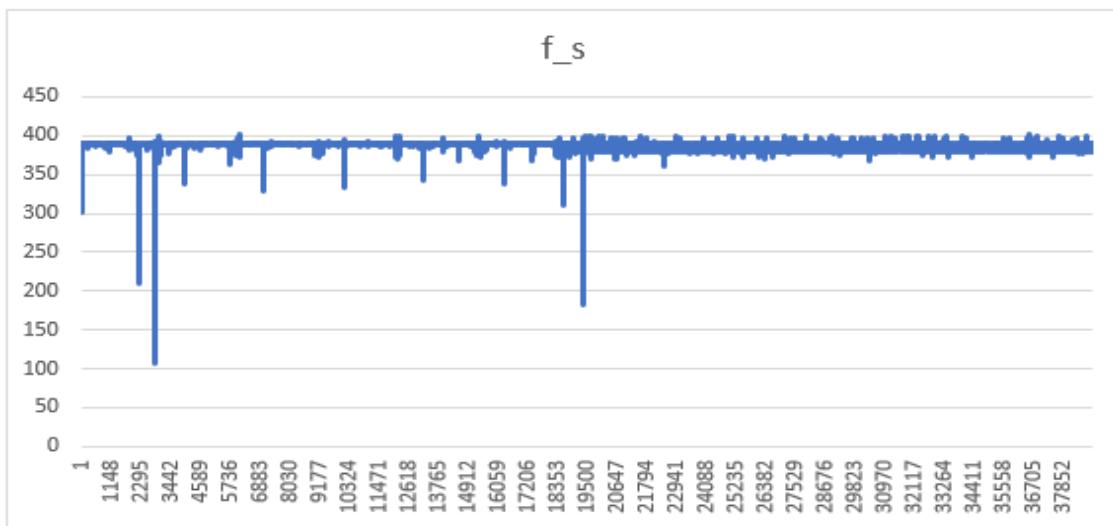


Figura 108: Variación de la frecuencia de muestreo en función de las muestras.

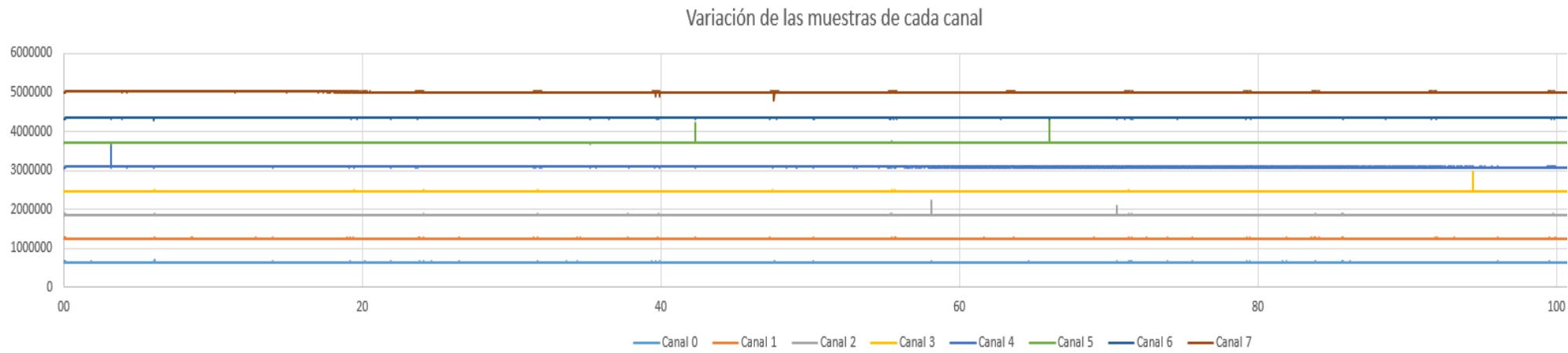


Figura 109: Variación temporal de las señales obtenidas por cada canal.

Adicionalmente se realizaron los graficos de la variación temporal en función de las muestras (figura 106), la variación del periodo de muestreo en función de las muestras (figura 107), la variación de la frecuencia de muestreo en función de las muestras (figura 108) y, por último, la variación de las mediciones de cada canal en función del tiempo.

Puede observar que eventualmente aparecen picos positivos en la figura 107 y en la figura 108, indicando incrementos repentinos aleatorios del período de muestreo y decaimientos de la frecuencia de muestreo respectivamente. Estos picos pueden ser causados por el incremento de la temperatura, interferencias externas, pulsos perdidos o defectuosos del reloj de la placa, etc. Sin embargo son casos excepcionales y de muy baja ocurrencia, siendo aceptable para los estandares del dispositivo y los requisitos fijados.

En la figura 109 puede observar que las muestras tomadas por cada canal se mantienen prácticamente constantes a lo largo de todo el periodo de medición, con la excepción de picos muy poco frecuentes. Estos picos pueden deberse a defectos propios del dispositivo o de las señales de entrada, ya que a pesar de haber ingresado tensiones continuas en cada canal, el dispositivo no se encuentra perfectamente aislado de interferencias externas, defectos en las conexiones, en las baterías y componentes, etc.

- **Cálculo del error estándar**

La desviación estándar (SD) representa la variación en los valores de una variable respecto de la media, mientras que el error estándar de la media (Estándar Error of the Mean, SEM) representa la dispersión que tendría la media de un conjunto de valores obtenidos si se continuaran tomando muestras. Por lo tanto, el SEM proporciona una idea de la precisión de la media y el SD nos da una idea de la variabilidad de las observaciones individuales. Estos dos parámetros están relacionados:

$$SEM = \frac{SD}{\sqrt{n}}$$

Donde:

SEM = Error estándar de la media

SD = Desviación estándar

n = tamaño de la muestra

En base a las muestras tomadas anteriormente, con una población de 39000 muestras, el cálculo del error estándar para cada canal es:

	Media canal 7	Error estándar canal 7
Cuentas	5014367.83	10.552564
Volts	2.988796	6.289818 uV
Porcentaje		2.104465*10^(-4) %

	Media canal 6	Error estándar canal 6
cuentas	4342568.40	7.581694
volts	2.588372	4.519041 uV
porcentaje		1.745901*10^(-4) %

	Media canal 5	Error estándar canal 5
cuentas	3709739.59	22.358054
volts	2.211177	13.326439 uV
porcentaje		6.026853*10^(-4) %

	Media canal 4	Error estándar canal 4
cuentas	3085619.06	17.019632
volts	1.839172	10.144491 uV
porcentaje		5.515793*10^(-4) %

	Media canal 3	Error estándar canal 3
cuentas	2469698.20	13.063106
volts	1.472055	7.786217 uV
porcentaje		5.289352*10 ⁻⁴ %

	Media canal 2	Error estándar canal 2
cuentas	1863056.46	11.013398
volts	1.110468	6.564497 uV
porcentaje		5.911468*10 ⁻⁴ %

	Media canal 1	Error estándar canal 1
cuentas	1254268.97	2.284809
volts	0.747603	1.361852 uV
porcentaje		1.821624*10 ⁻⁴ %

	Media canal 0	Error estándar canal 0
cuentas	644480.15	2.132785
volts	0.384140	1.271239 uV
porcentaje		3.309312*10 ⁻⁴ %

▪ Contrastación del instrumento

Para poder medir la exactitud, precisión y calidad de nuestro dispositivo de medición se lo debe comparar y contrastar con un instrumento de gran calidad.

El instrumento de referencia utilizado es un multímetro de banco UNI-T modelo UT803, el cual fue proporcionado por la escuela técnica 4-055 Pbro. Constantino Spagnolo y posee las siguientes especificaciones:

Pantalla		5999 dígitos, LCD, iluminación, 128 x 28 mm
DC voltaje	Diapasón	600 mV / 6 V / 60 V / 600 V / 1000 V
	Precisión	±(0.3%+2)
AC voltaje	Diapasón	600 mV / 6 V / 60 V / 600 V / 1000 V
	Precisión	±(0.6%+5)
DC corriente	Diapasón	600 µA / 6000 µA / 60 mA / 600 mA / 10 A
	Precisión	±(0.5%+3)
AC corriente	Diapasón	600 µA / 6000 µA / 60 mA / 600 mA / 10 A

	Precisión	$\pm(1\%+5)$
Resistencia	Diapasón	600 Ω / 6 k Ω / 60 k Ω / 600 k Ω / 6 M Ω / 60 M Ω
	Precisión	$\pm(0.5\%+2)$
Capacitancia	Diapasón	6 nF / 60 nF / 600 nF / 6 μ F / 60 μ F / 600 μ F / 6 mF
	Precisión	$\pm(2\%+5)$
Frecuencia	Diapasón	6 kHz / 60 kHz / 600 kHz / 6 MHz / 60 MHz
	Precisión	$\pm(0.1\%+3)$
Temperatura	Diapasón	-40 °C ~ 1000 °C
	Precisión	$\pm(1\%+3)$
Interfaces		USB, RS232C
Alimentación		AC 220 V / 50 Hz o baterías de 1.5 V (R14) x6

Como puede observarse posee una exactitud en la medición de tensión continua de $\pm(0.3\% + 2$ dígitos) y en CA de $\pm(0.6\% + 5$ dígitos) y los rangos de medición son mucho mayores a los requeridos para las mediciones a realizar, por lo que será más que suficiente para realizar la contrastación.

Para ello se medirá y graficará la dispersión de las mediciones respecto de la media para 3 valores distintos y la linealidad tomando 10 valores en forma ascendente.

Se deben ingresar las mismas señales a ambos instrumentos en el mismo instante y tomar las mediciones en forma simultánea para una correcta contrastación.

De nada sirve poseer un instrumento de gran precisión si las condiciones en las que se realizan las mediciones no son adecuadas. Hay muchos factores externos que afectan significativamente la exactitud de las mediciones:

- Interferencias externas sobre los cables, sobre las entradas de medición y sobre los componentes del equipo bajo prueba y de las propias placas. Para reducir y minimizar su efecto se utilizaron cables coaxiales, los cuales poseen el apantallamiento necesario para minimizar el efecto de las interferencias, ya que poseen un mallado externo que conectaremos a la tierra del circuito.
- También se utilizó blindaje para el circuito y la placa. Los mismos fueron colocados dentro de un rescinto metálico conectado a tierra, el cual deriva todas las corrientes inducidas por interferencias a tierra. Además, esto minimiza tanto el efecto de la luz que también es una fuente de interferencias, como la carga electrostática que poseen las personas y afectan las mediciones al acercarse al circuito.

- También se debe disminuir al mínimo la longitud de los terminales de los componentes y las conexiones , ya que actuan como pequeñas antenas para señales interferentes de radiofrecuencia.
- Por ultimo, no es posible verificar que la dispersión de las mediciones se debe justamente al propio instrumento o a la variacion del voltaje de referencia que se esté utilizando. Por esto, la fuente de tensión debe ser lo mas estable posible. La mejor alternativa es el uso de baterias que poseen gran estabilidad de tensión en el corto tiempo. Por esto las mediciones deben hacerse dentro de los 10 o 15 segundos, para evitar que la descarga paulatina de la batería incida sobre las mediciones obteniendo falsas conclusiones.

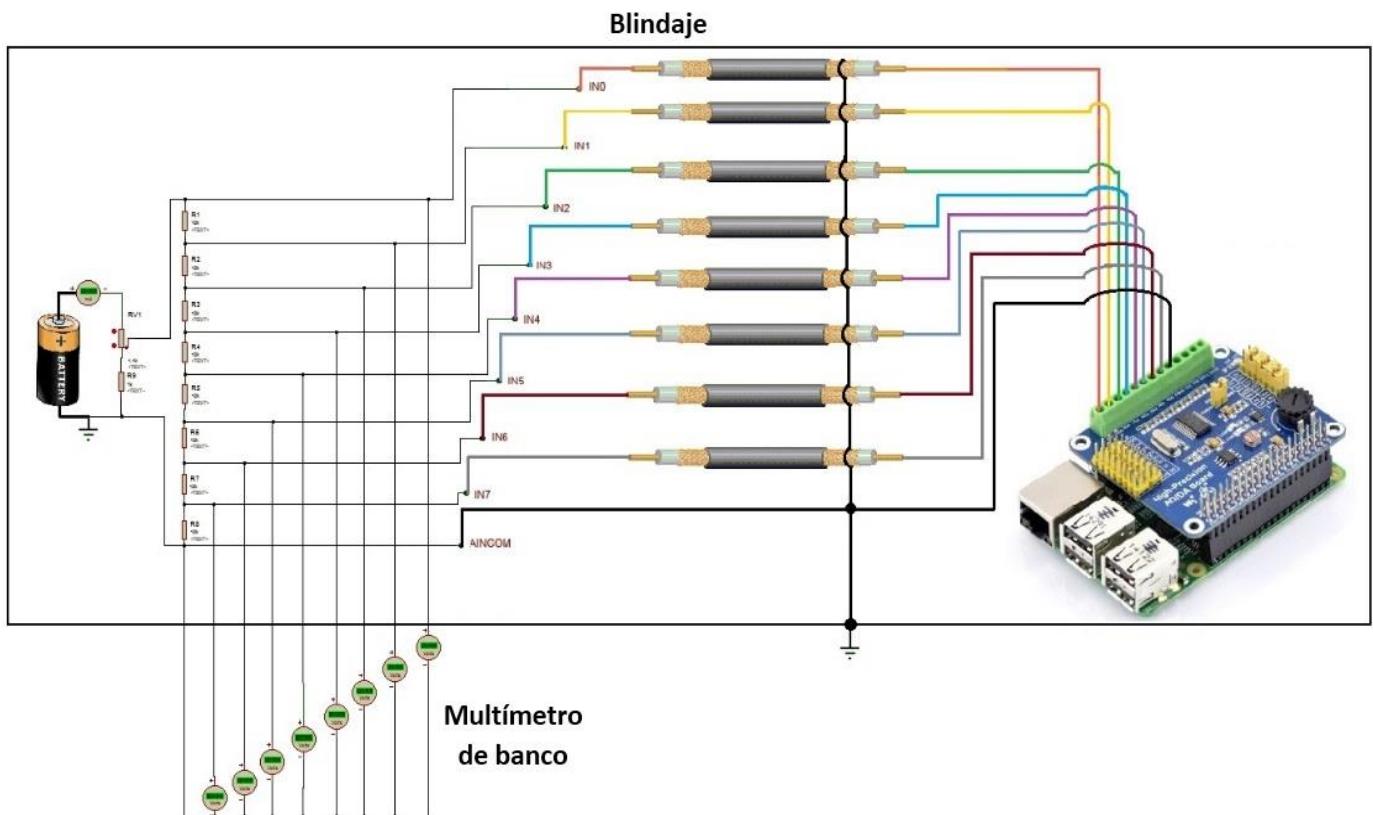


Figura 110: Montaje de medición para contrastación de instrumentos.

En este esquema los voltmetros conectados en paralelo a las entradas del conversor A/D, montado sobre la placa Raspberry Pi, representan al multímetro de banco utilizado como referencia.

Se tomaron muestras cada 1 segundo durante 15 segundos con el multímetro de banco. Al mismo tiempo se tomaron muestras durante los mismos 15 segundos con el conversor ADC a una frecuencia de 390 SPS, lo que dió un total de 5800 muestras.

Las mediciones, resultados y gráficos obtenidos fueron los siguientes:

Mediciones realizadas con el multímetro de banco:

Canal																
Tiempo (S)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.438	0.437	0.437	0.436	0.436	0.436	0.438	0.436	0.437	0.438	0.437	0.438	0.438	0.439	0.44	0.439
1	0.827	0.824	0.824	0.822	0.823	0.824	0.823	0.823	0.823	0.823	0.823	0.823	0.821	0.823	0.823	0.822
2	1.243	1.211	1.190	1.215	1.261	1.215	1.180	1.190	1.193	1.191	1.194	1.240	1.224	1.222	1.209	1.204
3	1.496	1.494	1.495	1.495	1.496	1.498	1.500	1.501	1.504	1.514	1.520	1.537	1.538	1.530	1.520	1.512
4	1.876	2.055	1.995	1.977	1.981	1.997	2.012	1.960	1.956	1.961	1.958	1.990	2.030	2.010	2.047	1.986
5	2.318	2.356	2.413	2.239	2.239	2.240	2.250	2.251	2.341	2.314	2.253	2.279	2.257	2.266	2.271	2.284
6	2.611	2.612	2.611	2.612	2.631	2.612	2.611	2.612	2.612	2.611	2.612	2.612	2.610	2.613	2.612	2.614
7	2.988	2.988	2.987	2.988	2.988	2.987	2.988	2.988	2.987	2.988	2.987	2.988	2.990	2.988	2.999	2.960

Los valores medios para cada canal son:

Canal 0	
Vmedio ADC	0.421558
Vmedio Volt	0.4375

Canal 1	
Vmedio ADC	0.807300
Vmedio Volt	0.823188

Canal 2	
Vmedio ADC	1.221979
Vmedio Volt	1.211

Canal 3	
Vmedio ADC	1.529774
Vmedio Volt	1.509

Canal 4	
Vmedio ADC	2.018756
Vmedio Volt	1.987

Canal 5	
Vmedio ADC	2.305651
Vmedio Volt	2.286

Canal 6	
Vmedio ADC	2.641513
Vmedio Volt	2.613

Canal 7	
Vmedio ADC	3.022286
Vmedio Volt	2.987

Canal	Delta	Error porcentual
0	0.015942	3.643913692
1	0.015888	1.930007978
2	0.010604	0.87533764
3	0.020399	1.351480061
4	0.031818	1.60136507
5	0.019963	0.87340914
6	0.028513	1.091216136
7	0.035473	1.187660224

El error porcentual fue calculado en referencia a las tensiones medidas por el multímetro de banco.

El gráfico comparativo de los valores medios de ambos instrumentos:

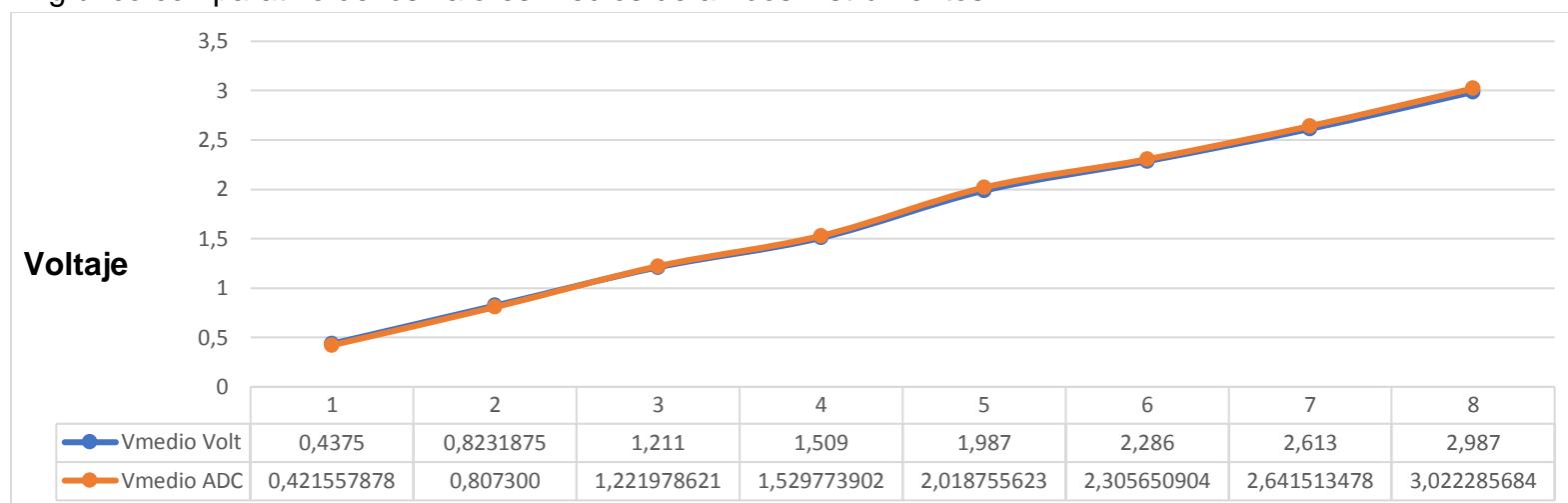
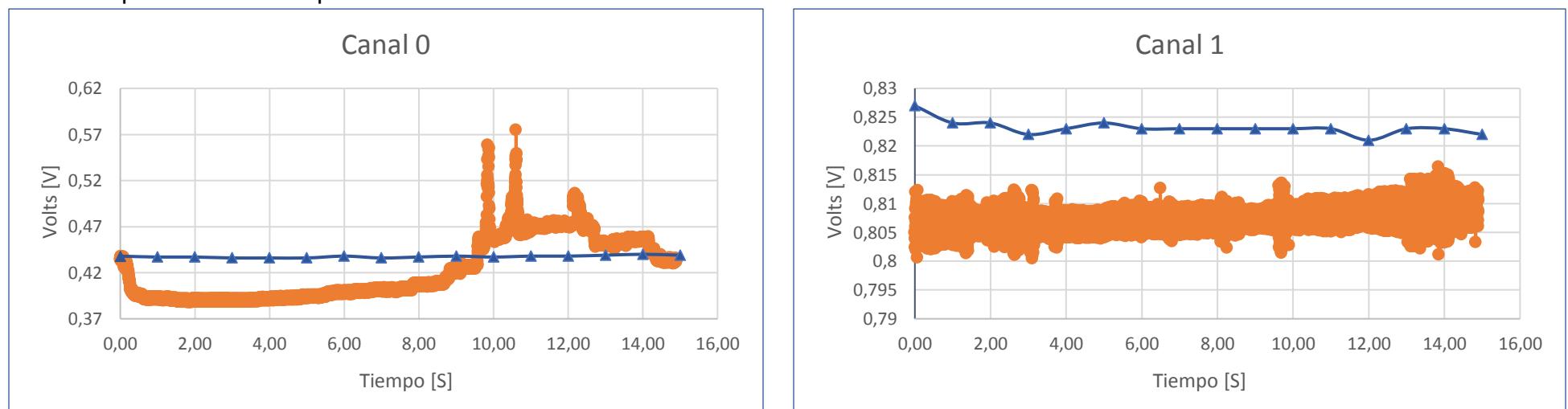
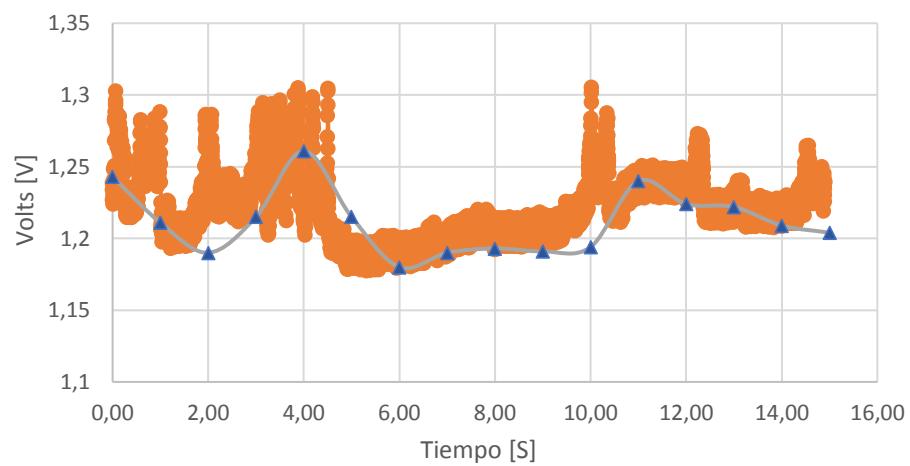


Figura 111: Comparación del promedio de muestras tomadas por ambos instrumentos.

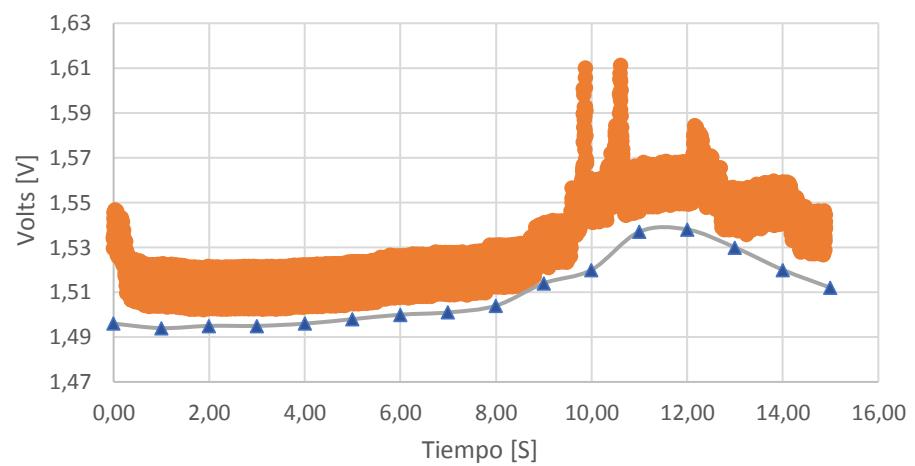
Y por último la comparación de las muestras instantáneas de cada canal:



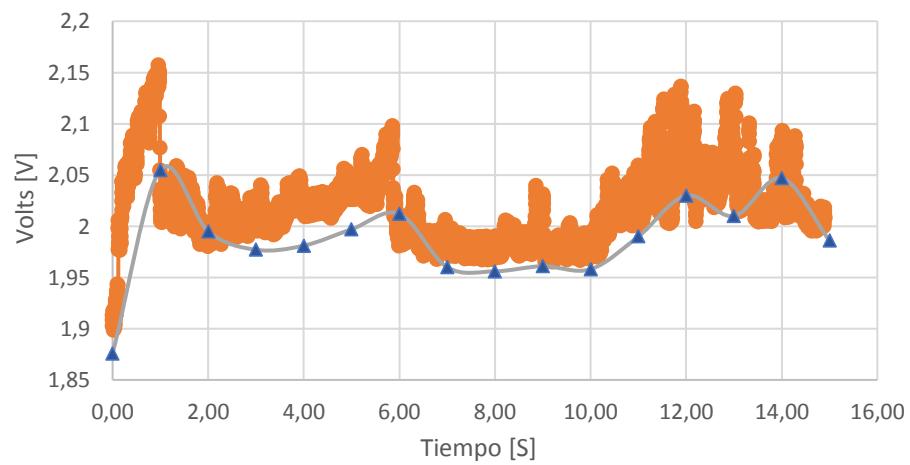
Canal 2



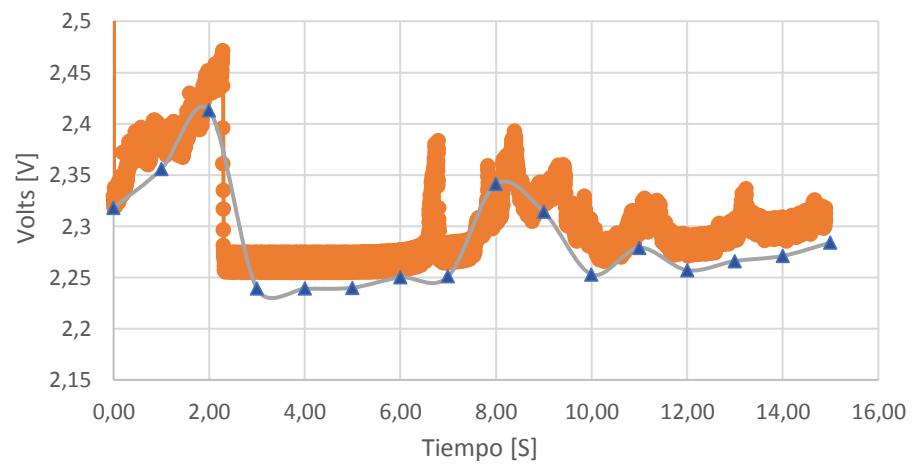
Canal 3



Canal 4



Canal 5



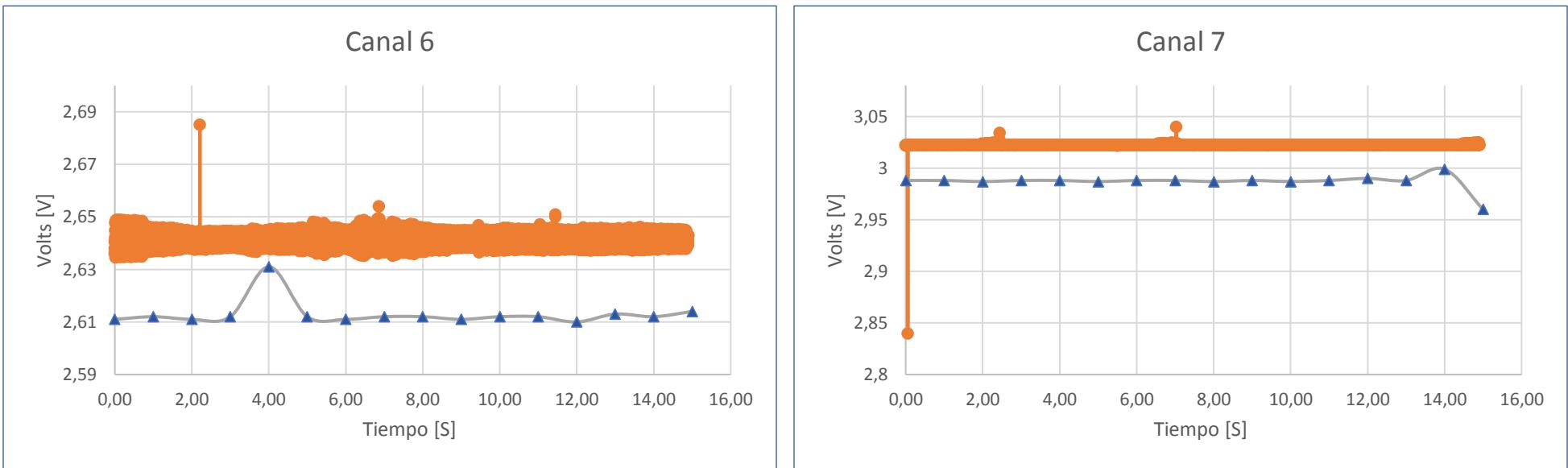


Figura 112: Variación temporal de las señales obtenidas por ambos instrumentos.

- **Linealidad:**

Para la determinación del grado de alinealidad del conversor, se ingresó una señal en forma de rampa a través de las entradas de este. Esta puede observarse a continuación:

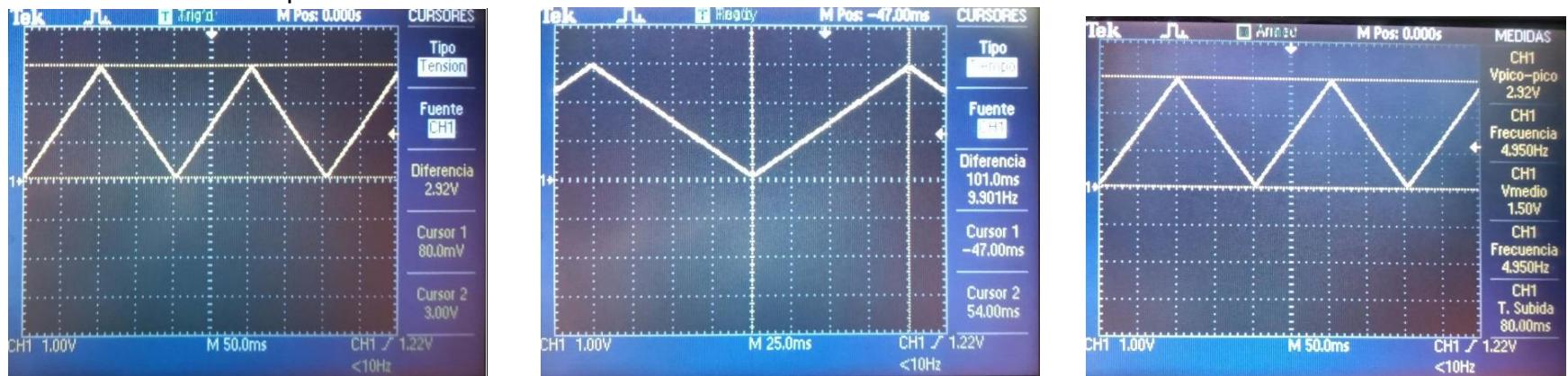


Figura 113: Señal triangular ingresada para determinar la linealidad.

Considerando solo el flanco de subida, obtenemos la recta de referencia:

$$\Delta V = 2,92 \text{ V} \quad \text{y} \quad \Delta T = 101 \text{ mS} \quad \longrightarrow \quad \text{Pendiente} = \Delta V / \Delta T = 28,9109 \text{ V/S}$$

$$V_{\text{initial}} = 80 \text{ mV}$$

$$\text{La ecuación de esta recta es: } V_{\text{rampa}} = (28,9109 \text{ V/S}) * t + 80 \text{ mV}$$

Esta rampa debe ser comparada con la rampa reconstruida por nuestro dispositivo y analizar la diferencia entre ambas.

La reconstrucción inicial de la rampa medida es:

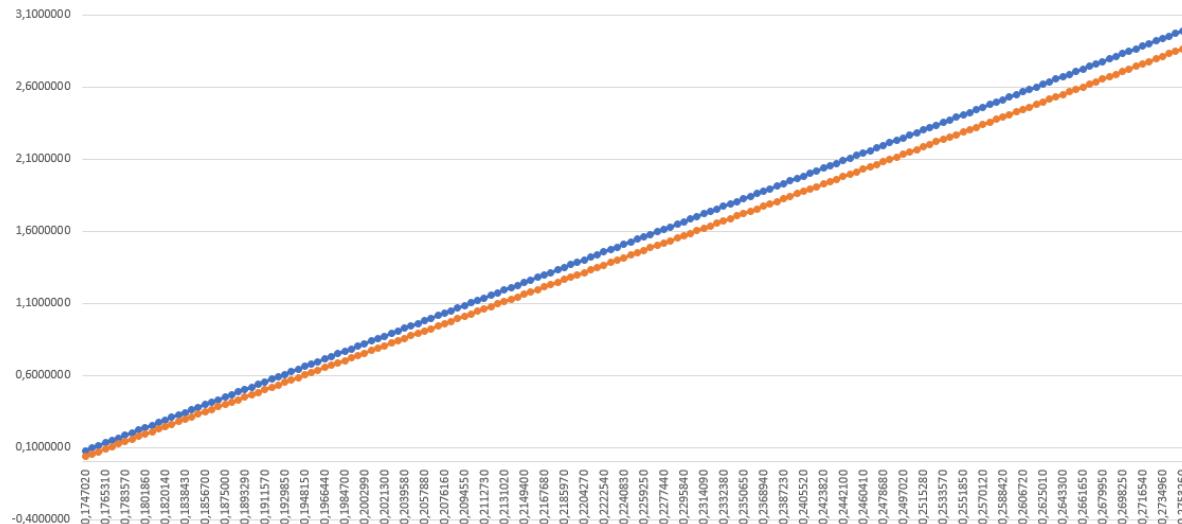


Figura 114: Comparación de la rampa ingresada y la rampa obtenida.

Siendo la recta azul la rampa ideal ingresada y la recta naranja la reconstruida. Puede observarse la diferencia existente entre ambas rectas, por lo que, aplicando un factor de corrección a la rampa reconstruida se minimiza el error cuadrático medio entre ambas rectas:

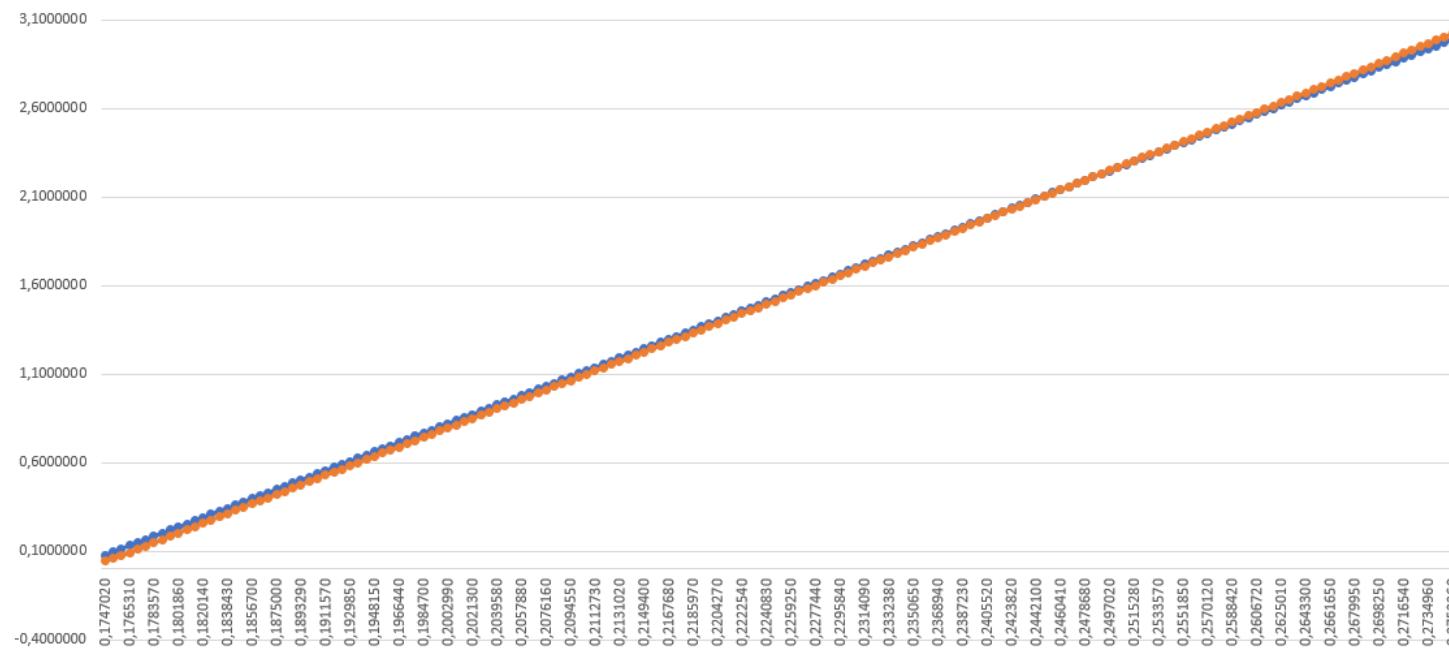


Figura 115: Comparación de la rampa ingresada y la rampa obtenida escalada.

El factor por el cual debe escalarse la rampa reconstruida para minimizar el RMSE es:

$$\text{Factor de corrección} = \mathbf{1,0547}$$

Siendo el error cuadrático medio igual a:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}} = \mathbf{0,02051381 \text{ o } 2,05138078 \% \text{ expresado porcentualmente.}}$$

Siendo P_i los valores de la recta ideal y O_i los valores de la recta reconstruida y escalada, para un tiempo t determinado y n la cantidad de muestras consideradas.

- **Repetibilidad entre canales:**

Debido a la conmutación entre canales de medición, aunque se aplique la misma señal en forma simultánea a todos los canales, existirá una leve diferencia entre los valores medidos entre canales. Esta diferencia será mayor mientras más grande sea la frecuencia y la rapidez con la que dicha señal cambie.

Se realizaron 3 pruebas con 3 señales senoidales de 3 frecuencias distintas, en el rango de tensión de 0.15 V a 2.85 V y frecuencias de uso cotidiano.

Se realizó el cálculo de la variación entre canales adyacentes tomando la diferencia entre las mediciones de los canales 0 y 1. También se calculó la variación entre los canales 0 y 7, los cuales poseen el mayor retardo entre sí al ser el primer y el último canal en medir respectivamente.

Por último se realizó el cálculo del error cuadrático medio entre canales adyacentes y entre el primer y último canal, para cada frecuencia.

- Señal de entrada de 1 Hz:

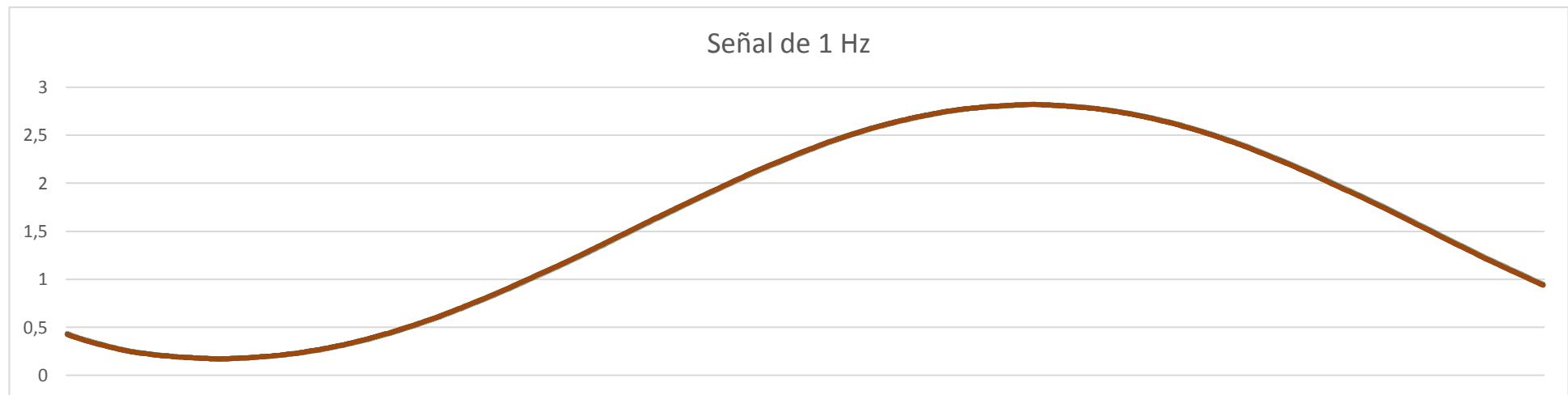


Figura 116: Reconstrucción de señal senoidal de 1 Hz.

Se han graficado las mediciones de todos los canales. Puede observar que no existe diferencia apreciable entre canales para una frecuencia tan baja.

Delta promedio canal 0 y 1 = 0.001231 V

Error cuadrático medio= 1.91286E-06

Delta promedio entre canal 0 y canal 7 = 0.008632 V

Error cuadrático medio= 9.35151E-05

- Señal de entrada de 5 Hz:



Figura 117: Reconstrucción de señal senoidal de 5 Hz.

En este caso puede observarse una leve diferencia entre las mediciones de los canales de entrada.

Delta promedio canal 0 y 1 = 0.008242 V

Error cuadrático medio= 8.24693E-05

Delta promedio entre canal 0 y canal 7 = 0.058133 V

Error cuadrático medio= 0.004109757

- Señal de entrada de 10 Hz:

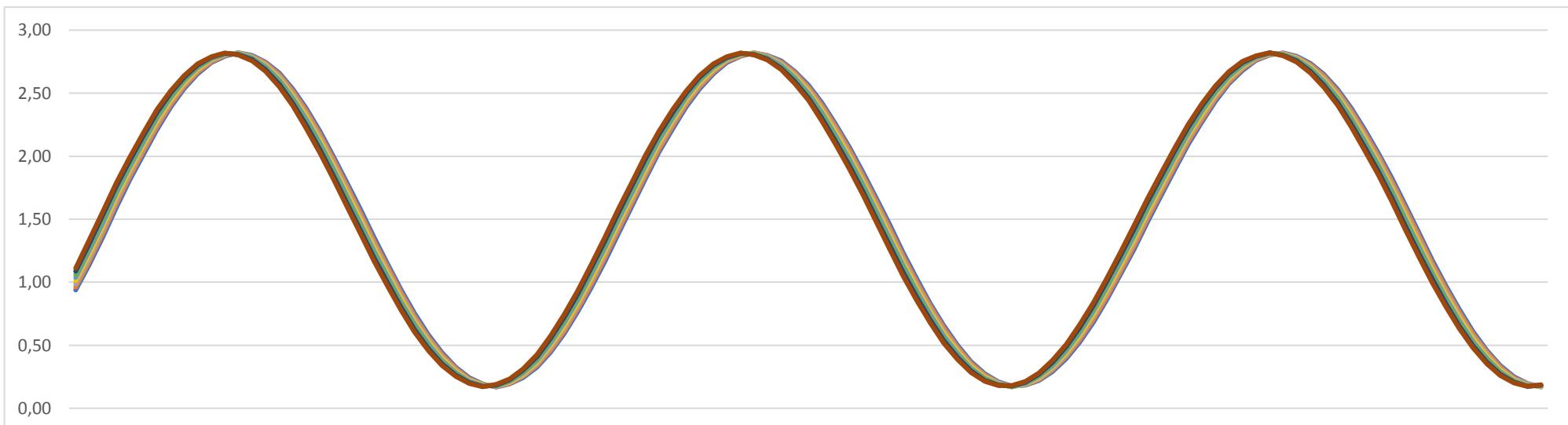


Figura 118: Reconstrucción de señal senoidal de 10 Hz.

Delta promedio canal 0 y 1 = 0.017071 V

Error cuadrático medio= 0.00036

Delta promedio entre canal 0 y canal 7 = 0.120128 V Error cuadrático medio= 0.017807

En conclusión, a medida que aumenta la frecuencia de la señal medida, más notoria es la diferencia entre los colores medidos, siendo la máxima entre los canales extremos 0 y 7 y en los flancos de subida y bajada donde la señal presenta una mayor variación de tensión.

- **Ancho de banda analógico**

Con el fin de determinar el límite superior de frecuencia de las señales que pueden ser ingresadas al conversor, se ingresó una señal cuadrada de baja frecuencia. Midiendo el tiempo de subida de dicha onda puede determinarse la frecuencia analógica superior para la cual la señal decae en 3 dB.

La señal ingresada es la siguiente:

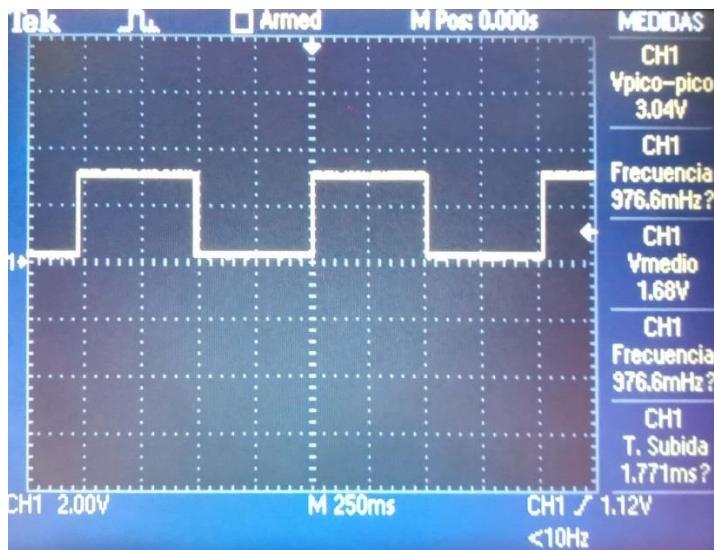


Figura 119: Señal cuadrada ingresada para medir el ancho de banda.

La señal reconstruida a partir de las muestras obtenidas, con una frecuencia de muestreo de aproximadamente 390 muestras/segundo es la siguiente:



Figura 120: Flanco de subida reconstruido a 390 muestras/segundo.

Puede observarse que el tiempo de subida se encuentra entre dos muestras consecutivas, por lo tanto, este será prácticamente igual al periodo de muestreo. Esto es debido a que la frecuencia de muestreo no es lo suficientemente elevada como para tomar muestras que se encuentren dentro del flanco de subida.

Esta frecuencia de muestreo es la que se posee cuando se comutan los 8 canales de medición. Por lo tanto, en estas condiciones y basándonos en el teorema de Nyquist, la máxima frecuencia que puede poseer la señal de entrada está limitada por la máxima frecuencia de muestreo, y será ésta dividida por dos, es decir, 195 Hz.

Con el fin de obtener un resultado más concluyente, se modificó el programa de medición para que solo midiera uno de los canales de medición. De esta forma se obtuvo una frecuencia de muestreo de aproximadamente 1642 muestras/segundo. La señal obtenida en este caso fue:

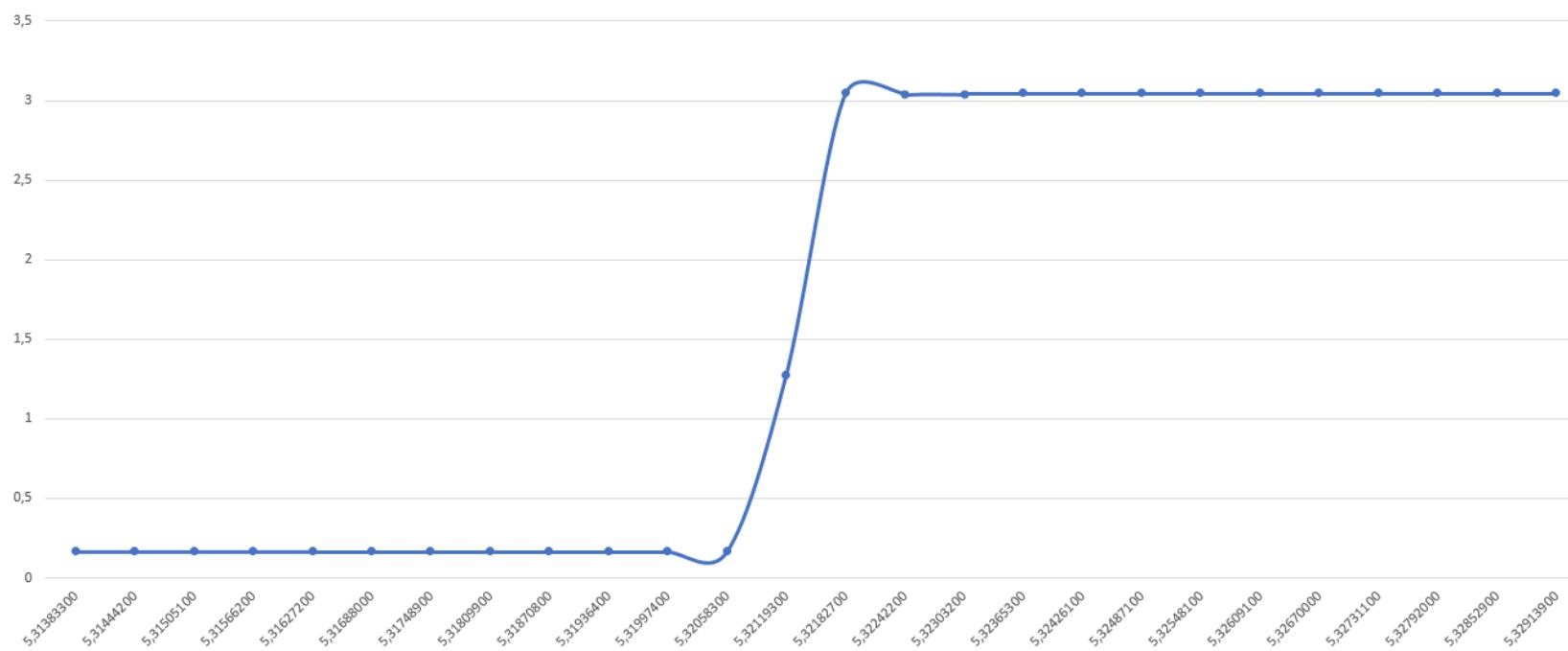


Figura 121: Flanco de subida reconstruido a 1642 muestras/segundo.

En este caso se alcanza a obtener una muestra a la mitad del flanco de subida, con lo cual podemos aproximar mejor la rampa obtenida.

Eliminando el offset de la señal y tomando los valores inferior y superior, podemos aproximarla a una rampa ideal:

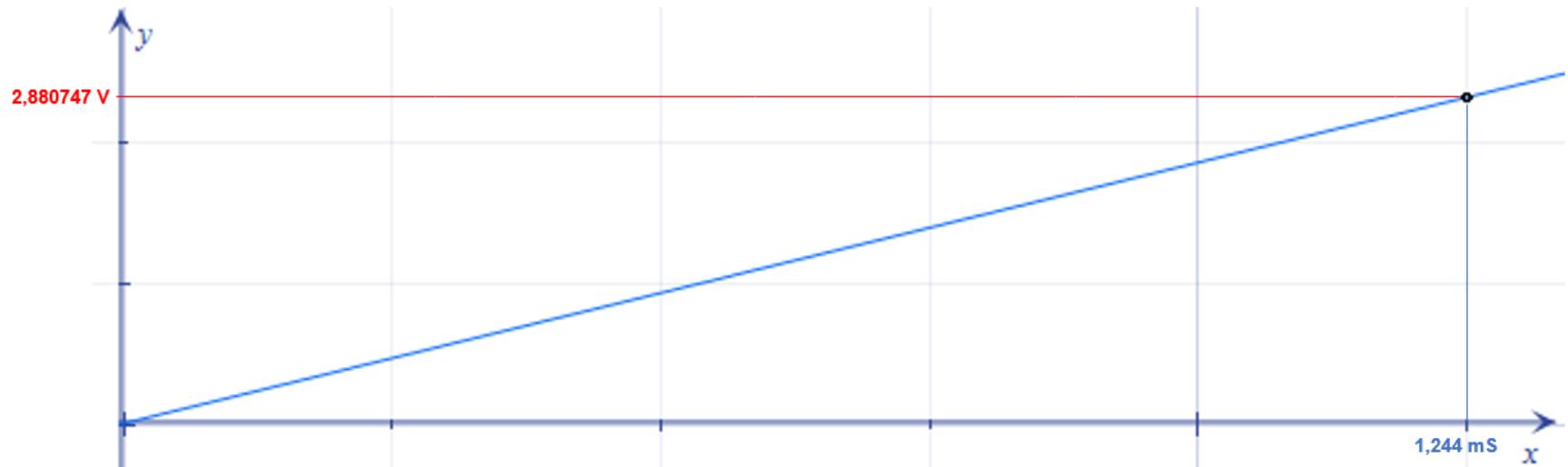


Figura 122: Rampa reconstruida ideal.

$$V_{rampa} = \left(2,3157128 \frac{V}{mS} \right) * t$$

El valor que toma esta rampa al 10 % de su valor máximo es:

$$V_{rampa}(10\%) = 2,880747 * 10\% = 0,2880747 V$$

Y el tiempo en que se alcanza este valor es:

$$Trampa(10\%) = 0,2880747 V / 2,3157128 (V/mS) = 0,1243999 mS$$

El valor que toma esta rampa al 90 % de su valor máximo es:

$$V_{rampa}(90\%) = 2,880747 * 90\% = 2,59267205 V$$

Y el tiempo en que se alcanza este valor es:

$$Trampa(90\%) = 2,59267205 V / 2,3157128 (V/mS) = 1,1195999 mS$$

Por lo tanto, el tiempo de subida es: $T_{subida} = Trampa(90\%) - Trampa(10\%) = 0,9952 mS$

Y la frecuencia límite superior es: $f_{sup} = 0,35/T_{subida} = 351,688 Hz$

- **Impedancia de entrada**

Para la obtención de las impedancias de entrada utilizamos un método de medición indirecto. A través de una resistencia conocida conectada en serie en la entrada de medición, se obtiene la diferencia de tensión en cada canal antes y después de colocar la resistencia auxiliar. Se realizaron los cálculos para cada canal con el buffer de entrada habilitado y deshabilitado.

- **Con buffer de entrada deshabilitado:**

Primero debe medirse la tensión de la pila conectada en forma directa a la entrada.

El resultado promedio en el canal 7 es de 4899274.589.

$$V_{\text{medio}} = 4962197.423 * 5 / 8388608 = 2.957700147 \text{ V}$$

Agregamos una resistencia de 81 KΩ (80800 Ω medidos con el multímetro de banco) en serie a la pila de la siguiente forma:

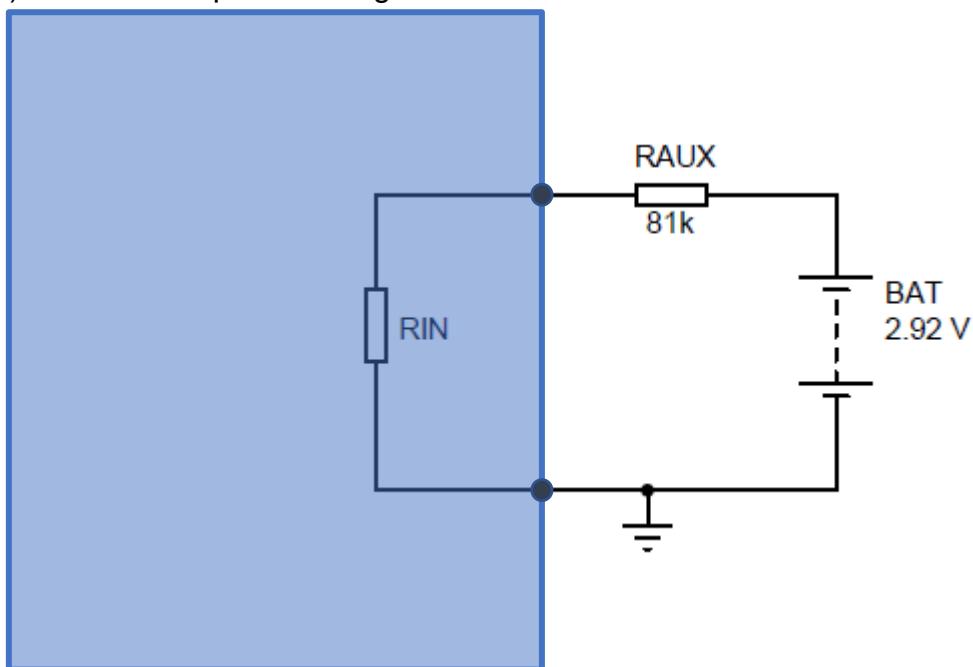


Figura 123: Circuito de prueba para medición de resistencia interna.

El resultado promedio en el canal 7 ahora es de 4670616.758.

$$V_{\text{medio}} = 4948685.966 * 5 / 8388608 = 2.949646691 \text{ V}$$

Por lo tanto, se tiene:

$$V_{\text{batería}} = I * R + V_{\text{in}}$$

$$2.957700147 \text{ V} = I * 80.8 \text{ K}\Omega + 2.949646691 \text{ V}$$

$$I = 9.94254E-08 \text{ A}$$

$$Z_{in} = V_{in}/I = 29666940.21 \Omega = 29.667 \text{ M}\Omega$$

Se realizó el mismo procedimiento con las demás entradas:

Canal	Valor medio	Vmedio [V]	Imedia [A]	R entrada [\Omega]
0	4951448.879	2.95129352	3.626E-08	81392077.16
1	4944081.491	2.94690221	9.0608E-08	32523626.87
2	4944955.354	2.94742307	8.4162E-08	35020945.49
3	4933377.775	2.9405223	1.6957E-07	17341323.7
4	4934495.213	2.94118834	1.6132E-07	18231536.4
5	4928992.442	2.93790844	2.0192E-07	14550070.55
6	4922466.602	2.93401873	2.5006E-07	11733398.43
7	4948685.966	2.94964669	9.9425E-08	29666940.21

También medimos la resistencia interna del multímetro de banco cuyas especificaciones dicen que posee una resistencia interna de 10 M\Omega.

$$V_{batería} = 2.954 \text{ V}$$

$$V_{in} = 2.932 \text{ V}$$

$$R = 80.8 \text{ K}\Omega$$

$$I = 2.72277*10^{-7} \text{ A}$$

$$R_{in} = 10.768436 \text{ M}\Omega$$

- **Con buffer de entrada habilitado:**

Una vez conocida la magnitud de las resistencias de entrada y sabiendo que la habilitación del buffer incrementará las resistencias de entrada, se cambió la resistencia auxiliar a 8.2 M\Omega, con el fin de obtener resultados más precisos.

Se modificó programa para habilitar el buffer analógico de entrada y se midió nuevamente la tensión de la pila sin la resistencia en serie:

El resultado promedio de la tensión de batería dio como resultado:

$$V_{medio} = 5157079.135 * 5 / 8388608 = 3.0738587 \text{ V}$$

Se agregó la resistencia en serie y se repitió el proceso:

Canal	Valor medio	Vmedio [V]	Imedia [A]	R entrada [\Omega]
0	4541528.538	2.70696195	4.4098E-08	61384909.09
1	4395382.883	2.61985235	5.4568E-08	48010720.14

2	4553268.501	2.71395952	4.3257E-08	62740190.04
3	4276624.772	2.549067	6.3076E-08	40412677.36
4	4440969.001	2.6470238	5.1302E-08	51596619.48
5	4461277.555	2.65912864	4.9847E-08	53345422.49
6	4514662.016	2.69094826	4.6023E-08	58469780.56
7	4420362.88	2.63474159	5.2778E-08	49920738.02

Puede notar que efectivamente las resistencias de entrada se incrementaron en varios megaohms, incluso duplicándose en ciertas entradas.

- **Memoria RAM utilizada y sistema de archivos:**

Uno de los aspectos a tener en cuenta es la ocupación de memoria. Al ser un dispositivo con memoria limitada, se deberá prestar especial cuidado a no saturar la memoria RAM de la placa, así como también la memoria microSD conectada a ella. Es por ello que se realizará un cálculo aproximado de la cantidad de memoria utilizada durante la generación de archivos y la cantidad máxima de archivos que pueden almacenarse en un directorio.

Para calcular cuanta memoria se ocupa por cada fila del archivo se deberá tener en cuenta que el arreglo de enteros donde se almacenan las muestras está declarado como arreglo de números enteros, por lo que cada celda ocupa 32 bits de memoria. También se guardan las estampas de tiempo del canal 0 y estas medidas se almacenan en un arreglo de 64 bits.

La elección de utilizar 64 bits para el arreglo de tiempo se debe a que los valores arrojados por la función `gettimeofday` van creciendo a medida que transcurre el tiempo y debido a que el tiempo actual arrojado se expresa en segundos transcurridos y microsegundos desde las 00:00:00 del 1 de enero de 1970, estos valores suelen ser muy elevados. Por otro lado, una buena alternativa para ahorrar memoria y utilizar arreglos de 32 bits es colocar una sola estampa de tiempo cuando ocurre el disparo, y las demás estampas referidas a esa estampa inicial. De esta forma se requerirá un solo casillero de 64 bits y los demás, al ser relativos a este casillero, tendrán valores pequeños por lo que se podrán utilizar 32 bits. La desventaja es que las estampas de tiempo, al ser relativas a la estampa principal, las estampas de tiempo de las muestras posttrigger serán positivas y las pretrigger serán negativas; esto implica realizar cálculos posteriores para hacer la diferencia con la estampa principal de cada una para los valores negativos y a cada estampa superior restarle la estampa principal. Esto ocupa mucho tiempo de procesamiento y alarga significativamente el tiempo de inactividad de medición .

De la forma implementada actualmente la cantidad de memoria ocupada por fila será:

$$\text{RAM ocupada} = (32 \text{ bits}) * (8 \text{ canales}) + (64 \text{ bits}) * (2 \text{ estampas [segundos y microsegundos]})$$

RAM ocupada = 384 bits = 48 bytes por fila

Puesto que la cantidad máxima de muestras almacenadas antes de comenzar a sobrescribirlas es de 39000, la memoria estimada total ocupada por los buffers de almacenamiento será de:

$$\text{RAM total ocupada} = (\text{48 bytes/ muestra}) * (\text{39000 muestras}) = \text{1872000 bytes}$$

$$\text{RAM total ocupada} = 1872 \text{ Kilobytes} = 1.872 \text{ Megabytes}$$

Para calcular la memoria total ocupada por cada archivo se debe multiplicar los 384 bytes por la cantidad de filas totales, lo que dependerá de la cantidad de muestras pre disparo y post disparo seleccionadas por el usuario.

También se debe sumar la primera fila que contiene el encabezado de cada columna, pero debido a que solo aparece una vez, su incidencia en la cantidad de memoria total ocupada es insignificante.

- **Sistema de archivos:**

Otro limitante a tener en consideración es la cantidad de archivos almacenables por directorio. Esto tiene incidencia directa sobre el parámetro de configuración “cantidad de archivos almacenables” modificable por el usuario.

La cantidad de archivos y subdirectorios que puede contener un directorio depende de el sistema de archivos empleado, siendo diferente para cada sistema operativo y distribución de ellos.

Para conocer qué sistema de archivos utiliza la distro de Linux, Raspbian, utilizamos los siguientes comandos:

```

pi@raspberrypi:~ $ df -Th
Filesystem      Type      Size  Used Avail Use% Mounted on
/dev/root       ext4      29G   7.2G  21G  27% /
/devtmpfs       devtmpfs  430M    0  430M  0% /dev
tmpfs          tmpfs     463M    0  463M  0% /dev/shm
tmpfs          tmpfs     463M   18M  445M  4% /run
tmpfs          tmpfs     5.0M   4.0K  5.0M  1% /run/lock
tmpfs          tmpfs     463M    0  463M  0% /sys/fs/cgroup
/dev/mmcblk0p6  vfat     253M   48M  205M  19% /boot
tmpfs          tmpfs     93M   4.0K  93M  1% /run/user/1000
pi@raspberrypi:~ $ lsblk -f
NAME FSTYPE LABEL UUID                                     FSAVAIL FSUSE% MOUNTPOINT
mmcblk0

└─mmcblk0p1
    vfat    RECOVERY
            0060-25CC
└─mmcblk0p2

└─mmcblk0p5
    ext4    SETTINGS
            b70760db-5bfb-40ec-92b6-2bc82f8efa19
└─mmcblk0p6
    vfat    boot  CF60-146C
            204.4M   19% /boot
└─mmcblk0p7
    ext4    root  01846025-4451-483b-8385-656927aa00ef  20.2G   25% /

```

Figura 124: Visualización de los sistemas de archivos de la Raspberry Pi.

Qué capacidad tiene cada sistema de archivos

SISTEMAS DE ARCHIVO	TAMAÑO MÁXIMO DE VOLUMEN	TAMAÑO MÁXIMO DE ARCHIVO
FAT32	4 GB	8 TB
NTFS	16 EiB (1,845^7 TB)	16 EiB (1,845^7 TB) teóricos En la práctica el límite es de entorno a 256 TB
EXFAT	16 EiB (1,845^7 TB)	64 ZiB (6,4^10 TB)
HFS+	8 EiB (9,223^6 TB)	8 EiB (9,223^6 TB)
APFS	8 EiB (9,223^6 TB)	16 EiB (1,845^7 TB)
EXT4	1 EiB (1,153^6 TB)	16 TiB (17,5921 TB)

El sistema de archivos utilizado para la memoria RAM es tmpfs. Se trata de un sistema de ficheros montado que utiliza memoria volátil. Por lo que los datos que pueda contener se pierden al reiniciar el equipo. Por lo general puede utilizar también el espacio de intercambio o SWAP, en las situaciones en que haya poca memoria volátil, generalmente RAM.

Por último, utiliza **Sysfs** que es un sistema de archivos virtual que proporciona el núcleo Linux v2.6. Sysfs exporta información sobre los dispositivos y

sus controladores desde el modelo de dispositivos del núcleo hacia el espacio del usuario y también permite configurar parámetros.

Nos enfocaremos en el ext4 que es el que utiliza para el almacenamiento en disco de los archivos, y nos da la limitación del tamaño máximo de archivos y la cantidad máxima de archivos por directorio.

ext4 es la última versión de la familia de sistemas de ficheros ext. Sus principales ventajas radican en su eficiencia (menor uso de CPU, mejoras en la velocidad de lectura y escritura) y en la ampliación de los límites de tamaño de los ficheros, ahora de hasta 16TB, y del sistema de ficheros, que puede llegar a los 1024PB (PetaBytes).

Las características principales del sistema ext4 son:

- Número máximo de archivos: $2^{32} - 1$ (4294967295)
- Número máximo de archivos por directorio: ilimitado
- Tamaño máximo del archivo: $2^{44} - 1$ bytes (16 TiB - 1)
- Tamaño máximo de volumen: $2^{48} - 1$ bytes (256 TiB - 1)

Según estas características la cantidad máxima de archivos por directorio es ilimitada, es decir, hasta que se llene la memoria. Por lo tanto, no existirán impedimentos en la cantidad de archivos por directorio, teniendo en cuenta que la memoria posee un límite que no debe sobrepasarse.

▪ **Pruebas adversas en la generación de archivos:**

Se sometió al dispositivo a diversas situaciones para analizar su comportamiento y arreglar aquellos que no sean adecuados:

- Cortar la alimentación del equipo antes y/o mientras está midiendo para determinar qué sucede con los archivos generados.

Si se interrumpe el programa antes de abrir el archivo de medición, no ocurren problemas. En la próxima ejecución retoma el valor que corresponde al archivo.

Si ya se encuentra en la situación donde se comienzan a sobrescribir los archivos, aunque se corte el programa, al volver a ejecutarlo, si el programa con el número que correspondía ya fue eliminado, entonces simplemente crea un nuevo archivo como si fuera la primera vez que se crea el archivo en cuestión.

Si el programa se interrumpe luego de crear el archivo de medición, pero antes de modificar el archivo lastfile con el número del siguiente archivo a crear, en la siguiente ejecución, al leer cuál fue el último archivo creado se obtendrá el mismo número que en el caso anterior, por lo tanto, sobrescribirá el último archivo.

Esta situación se solucionó abriendo el archivo justo después de modificar lastfile. Podría existir el problema de modificar el archivo lastfile con el próximo número de archivo y mientras que el archivo con el actual nunca se crea. Esto no

ocurre ya que al cerrarse el programa antes de cerrar el descriptor de archivo de lastfile, nunca se modifica su valor y mantiene su valor previo.

- Intentar borrar un programa que no existe.

En este caso la función system devuelve un valor distinto de 0. Simplemente ignorando este valor de retorno no se producen fallos en el programa, y se modifica el valor de RetCode para indicar que se produjo esta situación.

- Abrir y modificar un archivo mientras se está escribiendo.

En este caso no se producen fallos ni excepciones, pero toma los valores que existían cuando se abrió el archivo y no los modificados. Es decir, si al momento de abrir el archivo el parámetro de cantidad de muestras pretrigger era de 20 muestras y antes de leer este valor se modifica este parámetro a 50 y se guarda el archivo, el valor devuelto es 20 y no 50.

Respecto de los archivos de medición, si se interrumpe el programa luego de haber escrito las mediciones, pero antes de cerrar el descriptor de archivo las muestras quedan almacenadas, es decir no ocurren problemas. Por otro lado, si la interrupción se produce dentro de los bucles for mientras se está escribiendo en el archivo, el mismo queda en blanco sin ninguna medición ni los encabezados.

- Interrumpir la conexión a internet cuando se está midiendo.

Al cortar la conexión a internet una vez que el programa ya está siendo ejecutado, los próximos archivos generados sin conexión a internet conservan la hora correcta, por lo tanto, si el programa se inicia con internet no habrá problemas con el manejo del tiempo.

Cuando la placa se inicia sin conexión a internet mantiene la fecha de la última vez que estuvo encendida y no la fecha actual. Esto quiere decir que no posee un reloj interno en tiempo real y su configuración inicial de la fecha y la hora la obtiene de internet.

La solución a este problema fue implementada a través del módulo RTC.

- Interrumpir la alimentación del dispositivo mientras se tiene abierto el archivo de configuración o el archivo lastfile.

Tanto si se interrumpe el programa mientras se tiene abierto el archivo parámetros.txt como lastfile.txt, los mismos no se ven afectados. Lo mismo ocurre si se corta la alimentación.

Otra situación a tener en cuenta es qué sucede si el archivo lastfile fuera corrompido haciendo que su información deje de ser el número del archivo siguiente a ser generado. Si en lugar de este número se colocan símbolos o letras, el programa lo tomará como un valor nulo, es decir, la cuenta se reinicia en 0. Esto hará que se sobreescrbían los archivos, pero evita que el programa se cuelgue.

Otro problema que se planteó es que sucedería si lastfile.txt se elimina o queda inutilizado por alguna razón. Originalmente se abría el archivo en modo r+, lo que significa que se abre en modo lectura y escritura, pero el archivo ya debe existir. Por lo tanto, si se elimina ocurre un fallo de segmentación cuando se intenta leer un archivo que ya no existe. La solución a esto es crear el archivo si no existe. Para ello se pregunta primero si el puntero devuelto por *fopen* en modo r+ es nulo al intentar abrir el archivo. De ser el caso, significa que no pudo abrirse el archivo y por lo tanto no existe. Esto implica que debe abrirse en modo w+, que abrirá el archivo en modo escritura y de no existir lo creará.

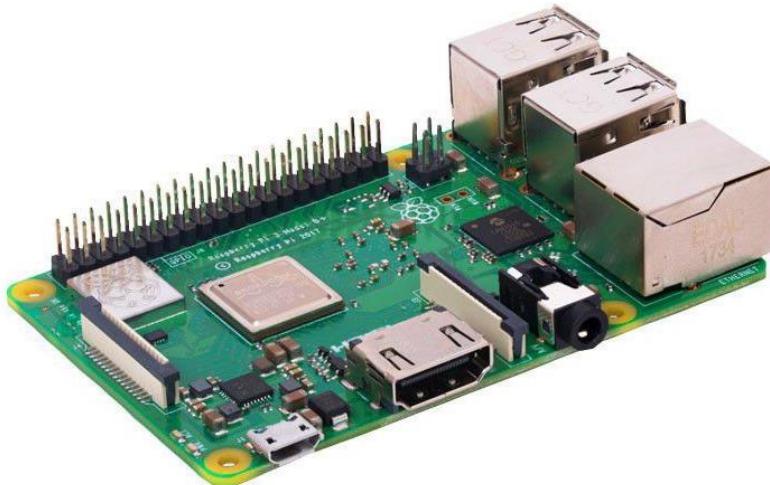
10 ANEXO: Manual de Instalación

El siguiente manual pretende guiar al usuario en el proceso de conexiónado, instalación y puesta en marcha del dispositivo desarrollado.

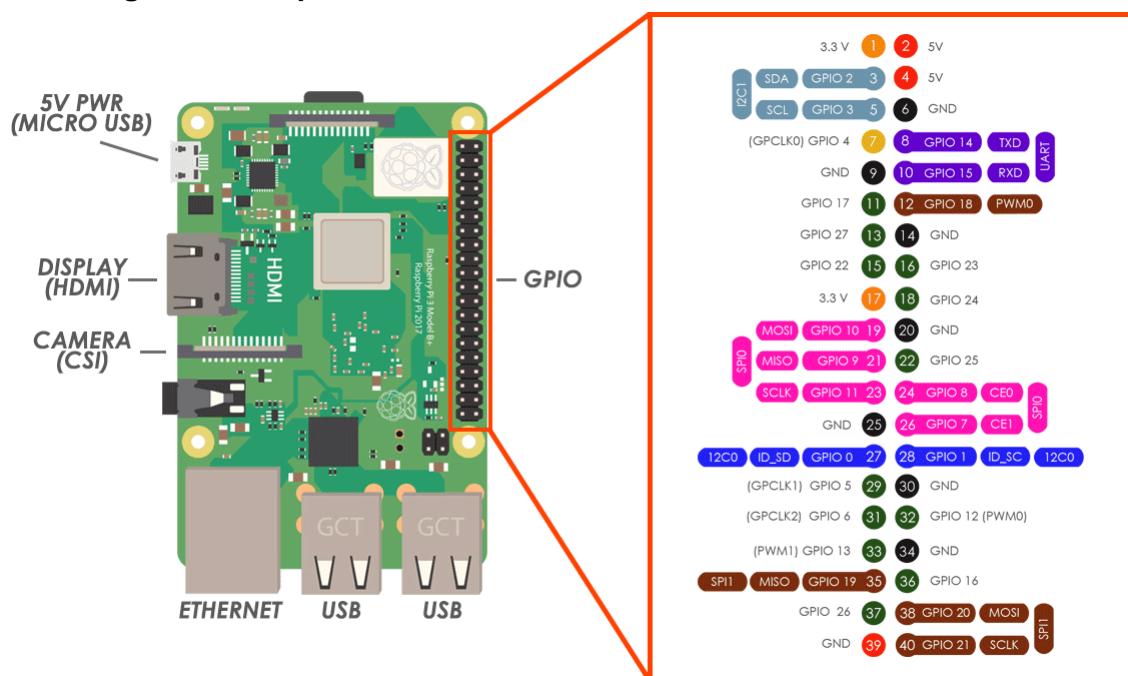
- **Paso 1: Conexión de los dispositivos.**

Se parte de 3 dispositivos básicos por separado: la placa Raspberry Pi modelo 3 B+, la placa conversora AD/DA de alta precisión y el módulo RTC. Se deberán conectar estos dispositivos entre sí.

- **Raspberry Pi 3 B+:**



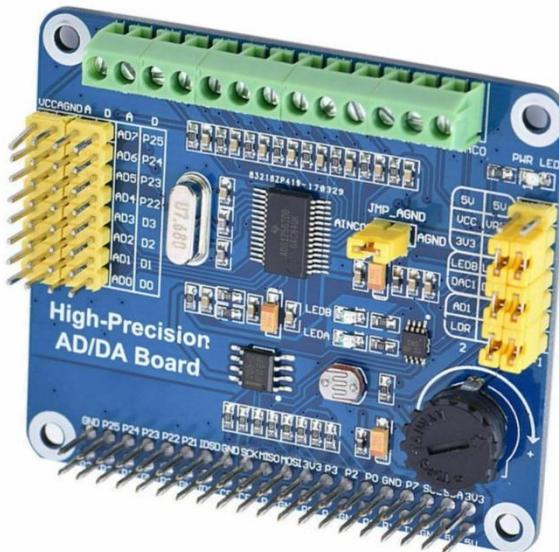
Configuración de pines:



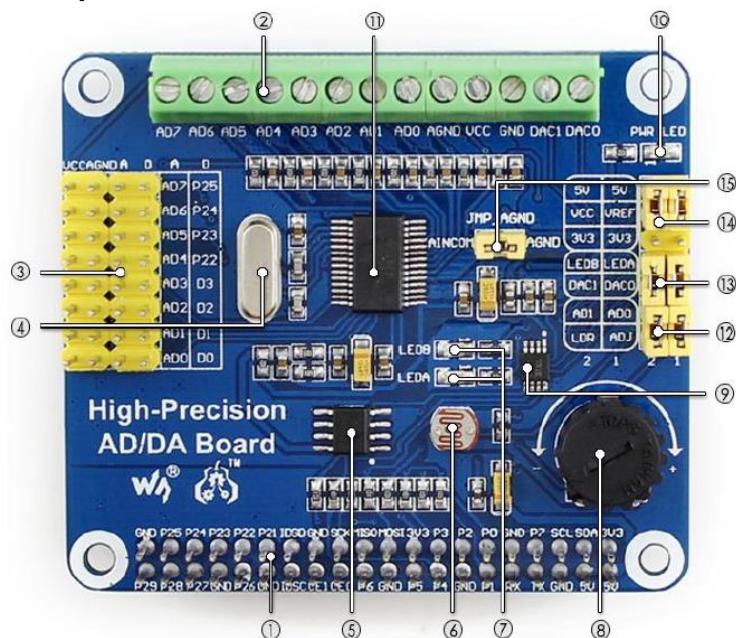
La alimentación es suministrada con un cargador de 5 V y 5.1 A, a través del conector micro USB.

Para el manejo de esta placa puede conectarse un teclado, mouse y monitor, gracias a sus 4 puertos USB y su conector HDMI. También puede accederse a internet a través de Ethernet, ya que posee un puerto RJ45.

- **Placa conversora AD/DA de alta precisión:**



Configuración de pines:

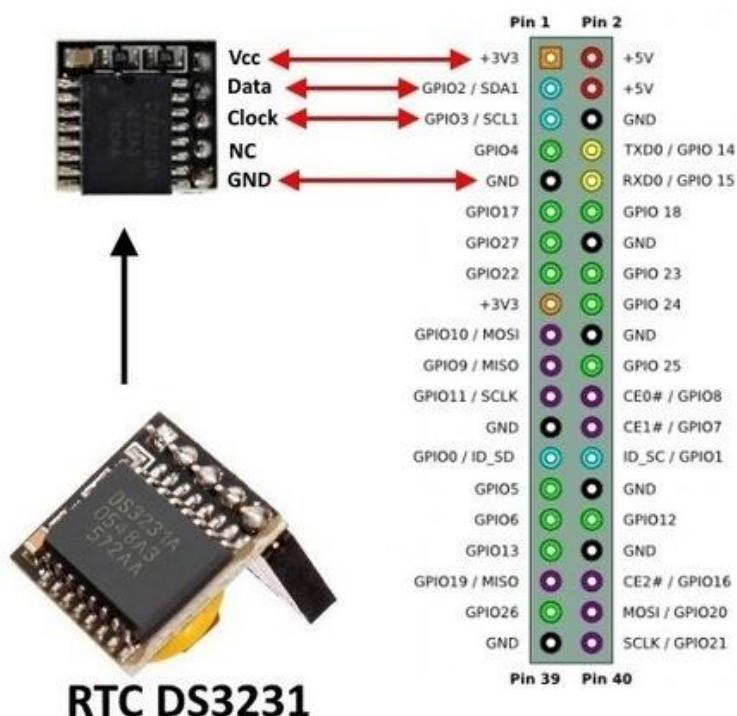


1. **Interfaz GPIO de Raspberry Pi:** para conexión con la placa Raspberry Pi.
2. **Entrada / salida AD / DA:** terminales de tornillo.
3. **Entrada AD:** cabezales de pin; los pinos de salida son compatibles con el estándar de interfaz del sensor Waveshare, fácil para conectar varios módulos de sensores analógicos.
4. **Cristal de 7,68 MHz.**
5. **LM285-2.5:** proporciona voltaje de referencia para el chip ADC.
6. **Foto resistor.**
7. **indicador de salida LED.**
8. **Potenciómetro de 10 KΩ.**

9. **DAC8532**: DAC de alta precisión de 16 bits, 2 canales.
10. **Indicador de alimentación**.
11. **ADS1256**: ADC de alta precisión de 24 bits, 8 canales (entrada diferencial de 4 canales)
12. **Puente de prueba ADC**.
13. **Puente de prueba DAC**.
14. **Puente de selección de potencia**. 21
15. **Configuración de tierra de referencia de ADC**: cuando se ingresa una sola entrada analógica, AINCOM es el terminal de referencia, el cual se puede conectar a GND o a un voltaje de referencia externo.

Los pines referenciados en 1 se conectan en forma directa sobre la placa Raspberry Pi.

- **Módulo RTC:**

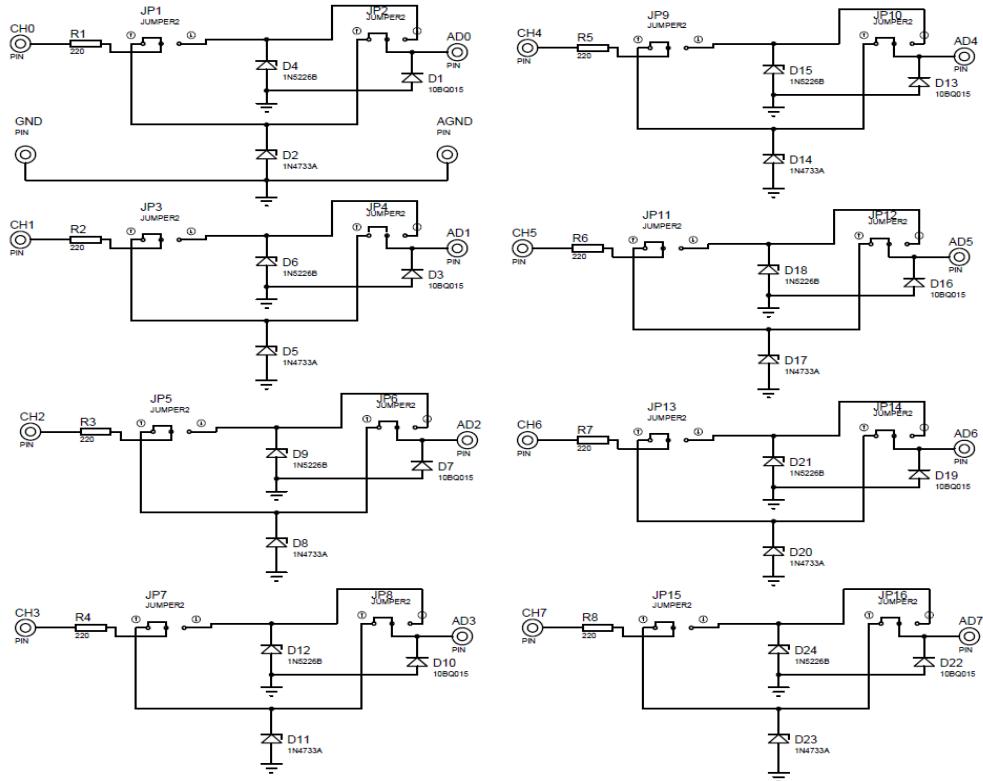


Los pines del módulo RTC se conectan sobre los pines de la placa conversora.

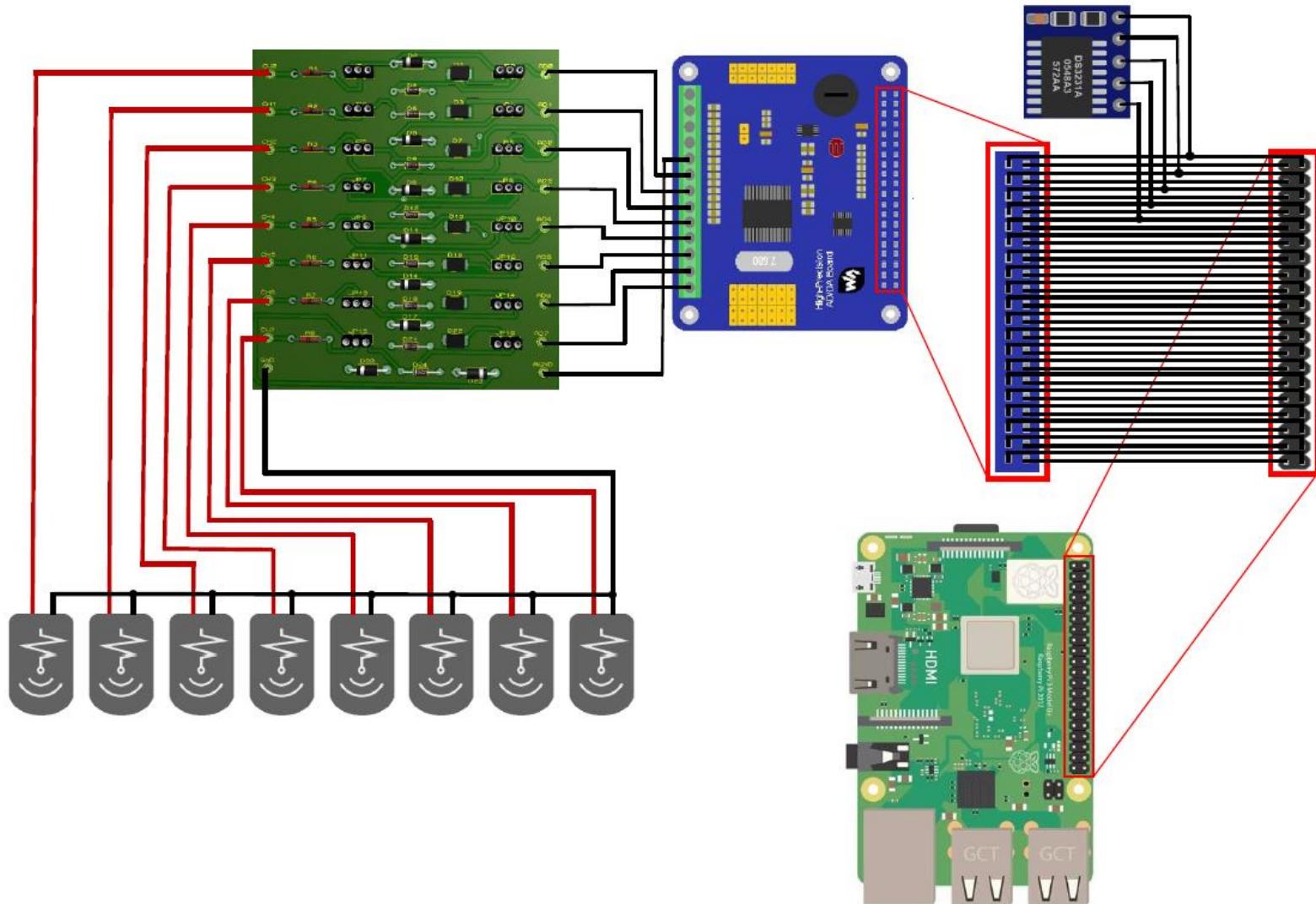
- **Circuito de protección:**

Este es el circuito encargado de proteger al dispositivo ante excesos en los límites de las señales de entrada. Por esto, los sensores de medición se conectan a las entradas de este circuito.

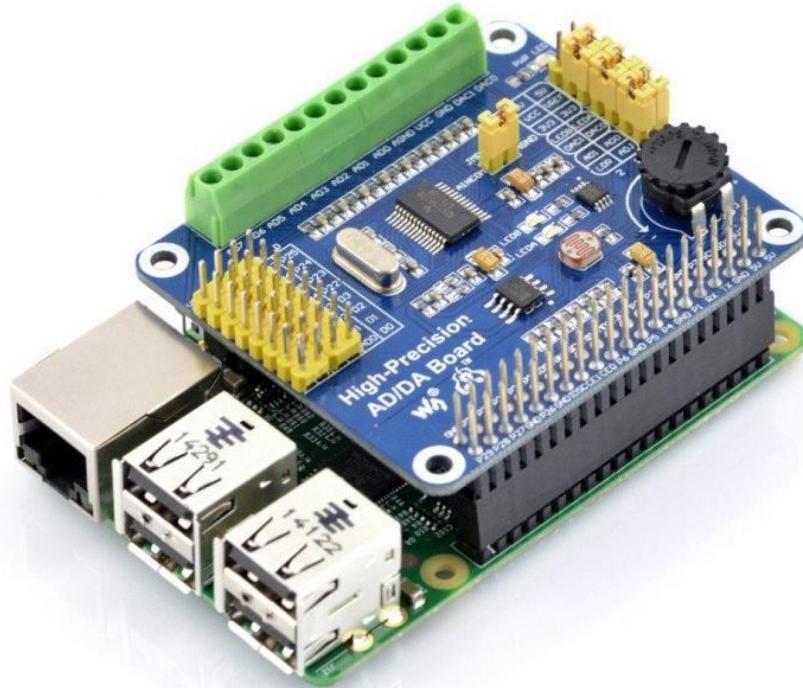
Un circuito esquemático y una vista preliminar de la placa pueden observarse a continuación:



Esquema de conexión completo:



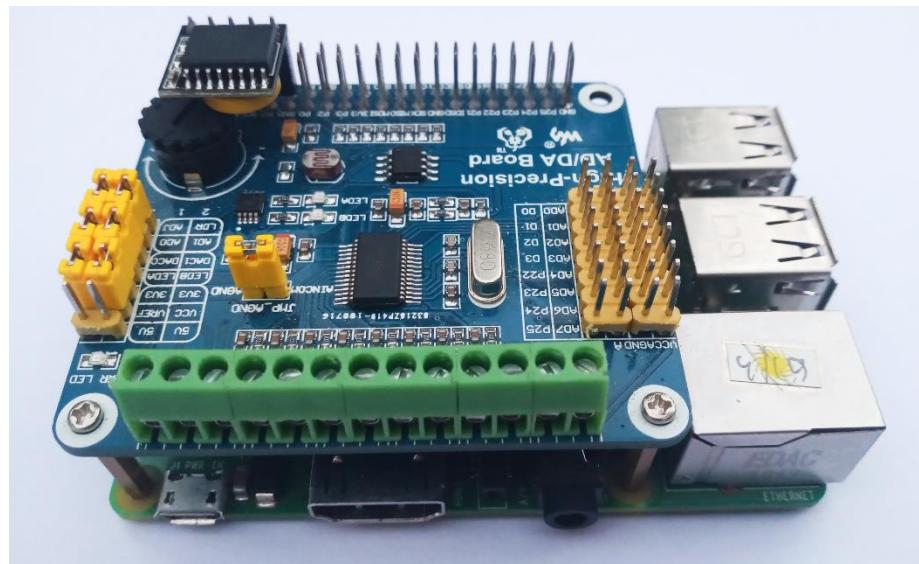
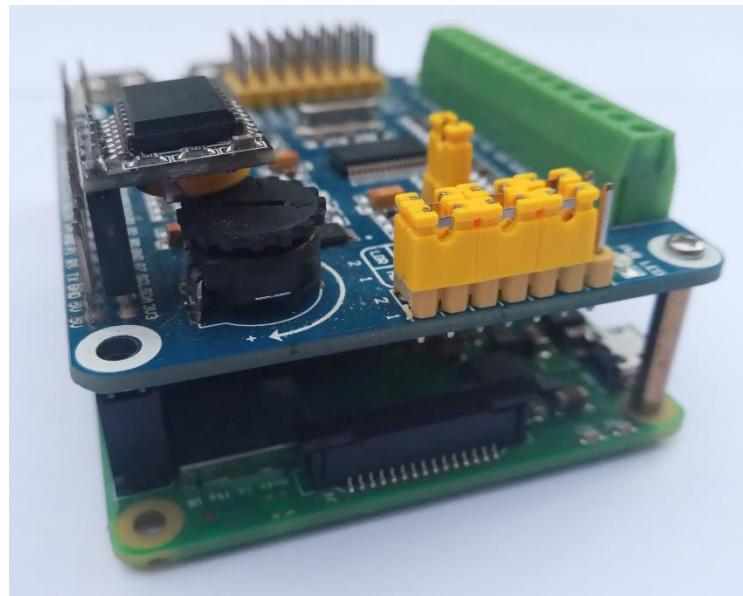
Monte primero la placa conversora sobre los pines de la Raspberry Pi:



Para una mejor sujeción entre las placas coloque tornillos pasantes en los orificios ubicados en las esquinas de la placa conversora.

Luego conecte el módulo RTC sobre la placa conversora en base a las conexiones especificadas. El montaje final (sin el circuito de protección) puede apreciarse en las siguientes figuras:





Coloque los puentes de la placa conversora de la siguiente manera:

- VCC a 5V
- VREF a 5V
- DAC0 a LEDA
- DAC1 a LEDB
- AD0 a ADJ
- AD1 a LDR

Esto se hace con jumpers en la placa en pines referenciados. Puede colocar VCC y VREF a los 3.3 V en vez de los 5 V, debido a que esta tensión de referencia es utilizada esencialmente por el DAC. Esta configuración implica que la salida DA0 se conecta al LED A, el cual se encuentra incluido en la placa, y la

salida DA1 al LED B. Por otro lado, la entrada AD0 se conecta a ADJ, el cual hace referencia al potenciómetro que viene incluido en la misma placa y AD1 se conecta al LDR, por lo que, modificando la posición del potenciómetro y la luz incidente sobre el LDR, modificamos los niveles de luz emitida por los leds A y B.

Precaución: la conexión entre los dispositivos debe hacerse con los mismos sin energía para evitar posibles falsos contactos que podrían dañar a los dispositivos.

- **Paso 2: Instalación de un sistema operativo.**

La Raspberry Pi es un ordenador de placa reducida. Es por estos que requiere de un sistema operativo para su correcto funcionamiento.

A continuación, se listan los más utilizados:

- **Raspbian**, es el sistema operativo oficial. Una distribución que funciona muy bien para una gran variedad de usos.
- **NOOBS y NOOBS Lite** son dos muy buenas opciones para principiantes en el uso de la Raspberry Pi.
- **Retroarch** una muy excelente distribución si lo que desea es convertir la Raspberry Pi en un emulador de consolas retro.
- **OSMC** un buen sistema operativo para crear un centro multimedia.
- **Windows IoT Core** es una propuesta algo más específica para instalaciones y control de dispositivos conectados.
- **Ubuntu** es el sistema clásico de las distros Linux y también se puede ejecutar en una Raspberry Pi si desea tener un equipo más funcional. Es ideal para las últimas versiones que cuentan con más RAM y potencia.

Se recomienda el uso del sistema operativo Raspbian, ya que, además de ser el sistema operativo oficial para el uso de placas Raspberry Pi es por lejos el más utilizado en la actualidad.

En específico para el desarrollo del dispositivo se utilizó Raspbian GNU/Linux 10 Buster.

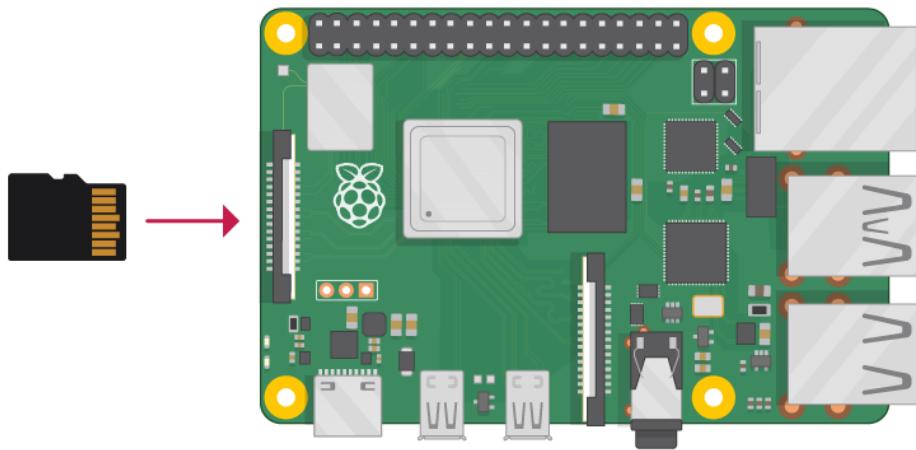
Los sistemas operativos de Raspberry Pi se instalan en tarjetas SD o MicroSD. Por tanto, requerirá una tarjeta con suficiente capacidad como para albergar el sistema operativo y los datos a almacenar. Entre 8 y 16 GB suele ser suficiente, aunque la mejor opción es una tarjeta MicroSD de 32 GB clase 10 para reducir al máximo los tiempos de lectura y escritura.

Los pasos de descarga e instalación de un sistema operativo puede seguirlos en los siguientes links:

- <https://www.raspberrypi.com/software/operating-systems/> para la descarga del sistema operativo.
- <https://eloutput.com/productos/domotica/instalar-so-raspberry-pi-herramientas/> para la instalación del sistema operativo.

Recomendación: previo a instalar el sistema operativo elegido, se recomienda el formateo de la tarjeta de memoria con sistema de archivos FAT32. Caso contrario pueden ocurrir problemas de corrupción de la tarjeta que obliguen nuevamente a formatear a esta con la posterior instalación del sistema operativo.

Una vez finalizados los pasos de instalación inserte la tarjeta MicroSD en la ranura correspondiente en la placa Raspberry Pi:



- **Paso 3: Acceso al dispositivo.**

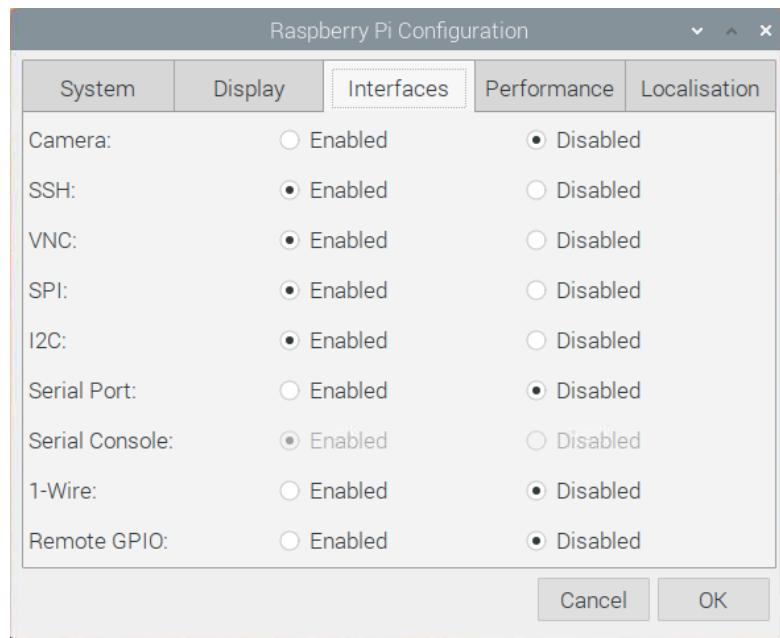
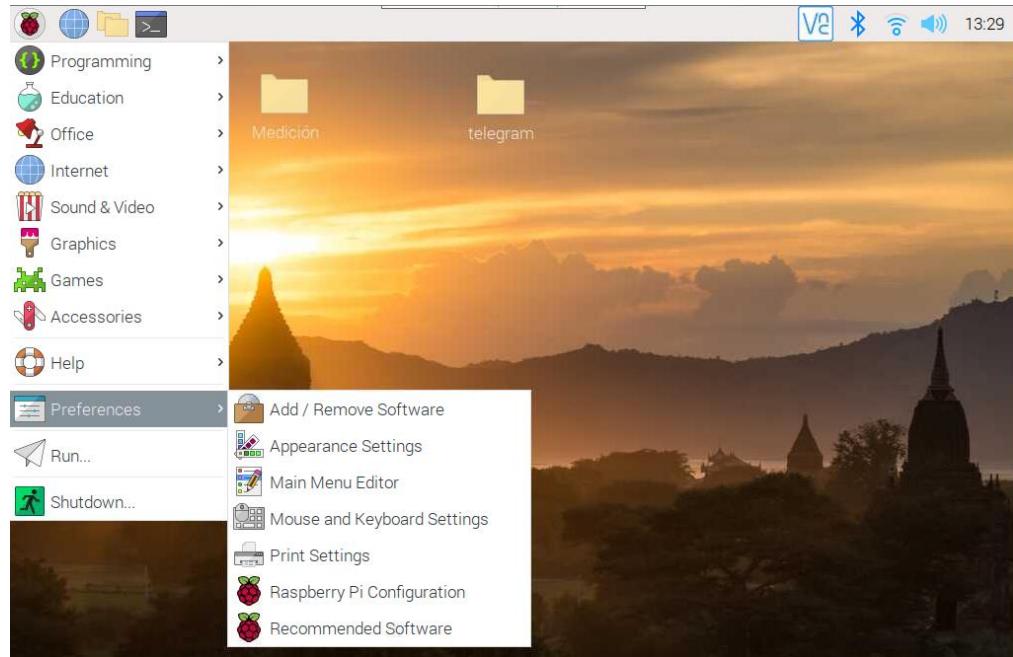
Una vez instalado el sistema operativo se deberá acceder a el para la posterior instalación de programas y configuración del dispositivo.

El acceso puede realizarse:

- **En forma directa:** conectando la placa Raspberry Pi a un mouse, teclado y monitor. Esta placa posee 4 puertos USB 2.0, un puerto HDMI y un puerto Ethernet para conexión a internet. Adicionalmente posee un puerto CSI para conectar una cámara y un puerto DSI para conectar una pantalla táctil. A su vez toma su alimentación a través de un adaptador 220VAC/5VDC conectado a la red eléctrica.
- **En forma remota:** ya sea a la línea de comandos o al escritorio de la placa.

Si desea hacerlo por el segundo método, requerirá conexión a internet, por lo que deberá conectarla a través del puerto Ethernet.

- I. **Habilitar comunicación SSH:** las placas Raspberry Pi traen por defecto desactivadas las conexiones SSH, por lo que no aparecerá un puerto asociado a ella. Esto ocasionará problemas al intentar conectarse remotamente. Por lo tanto, debe habilitarse. Esto se hace conectando la placa a un monitor y desde configuraciones de red habilitar dichas conexiones.



- II. Lo siguiente es averiguar la dirección IP y el puerto asignado a la placa. Para ello se recomienda el programa Nmap, el cual escanea todas las direcciones IP dentro de la red a través del envío de un ping. Para esto se ejecuta nmap desde la terminal de comandos a través del comando **nmap XXX.XXX.XXX.XXX/MM** colocando el nombre de su red con su respectiva máscara. Se arrojará un listado de los dispositivos conectados a la red:

```

C:\Users\Administrador>nmap 192.168.0.0/24
Starting Nmap 7.91 < https://nmap.org > at 2022-01-20 13:17 Hora est&ndash;ndar de Arg
entina
Nmap scan report for 192.168.0.1
Host is up <0.023s latency>.
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 58:D9:D5:28:C7:18 <Tenda Technology,Ltd.Dongguan branch>

Nmap scan report for 192.168.0.100
Host is up <0.020s latency>.
All 1000 scanned ports on 192.168.0.100 are closed
MAC Address: 08:A8:55:7B:9C:A3 <Motorola Mobility, a Lenovo Company>

Nmap scan report for 192.168.0.108
Host is up <0.0066s latency>.
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5900/tcp  open  vnc
MAC Address: B8:27:EB:4C:B3:17 <Raspberry Pi Foundation>

Nmap scan report for 192.168.0.112
Host is up <0.039s latency>.
Not shown: 995 closed ports
PORT      STATE SERVICE
3000/tcp  open  ppp
3001/tcp  open  nessus
7778/tcp  open  interwise
9080/tcp  open  gRPC
9998/tcp  open  distinct32
MAC Address: 30:A9:DE:25:50:D2 <LG Innotek>

Nmap scan report for 192.168.0.118
Host is up <0.020s latency>.
All 1000 scanned ports on 192.168.0.118 are closed
MAC Address: 8A:A8:9D:0B:32:10 <Unknown>

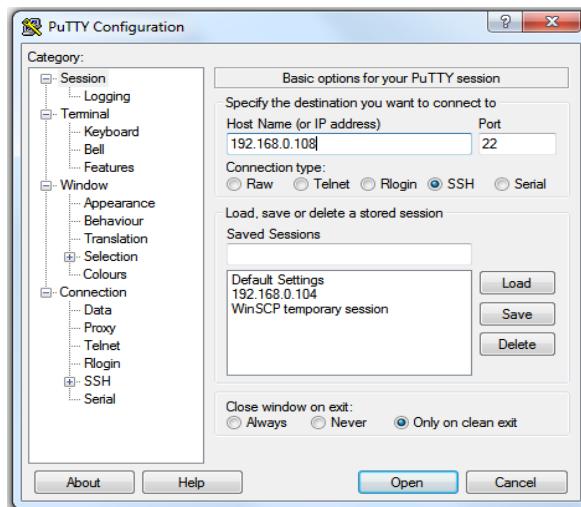
Nmap scan report for 192.168.0.110
Host is up <0.00018s latency>.
Not shown: 989 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1027/tcp  open  IIS
1028/tcp  open  unknown
2869/tcp  open  ictslap
3580/tcp  open  nati-svrloc
5357/tcp  open  wsddapi
8080/tcp  open  http-proxy

Nmap done: 256 IP addresses (6 hosts up) scanned in 8.07 seconds

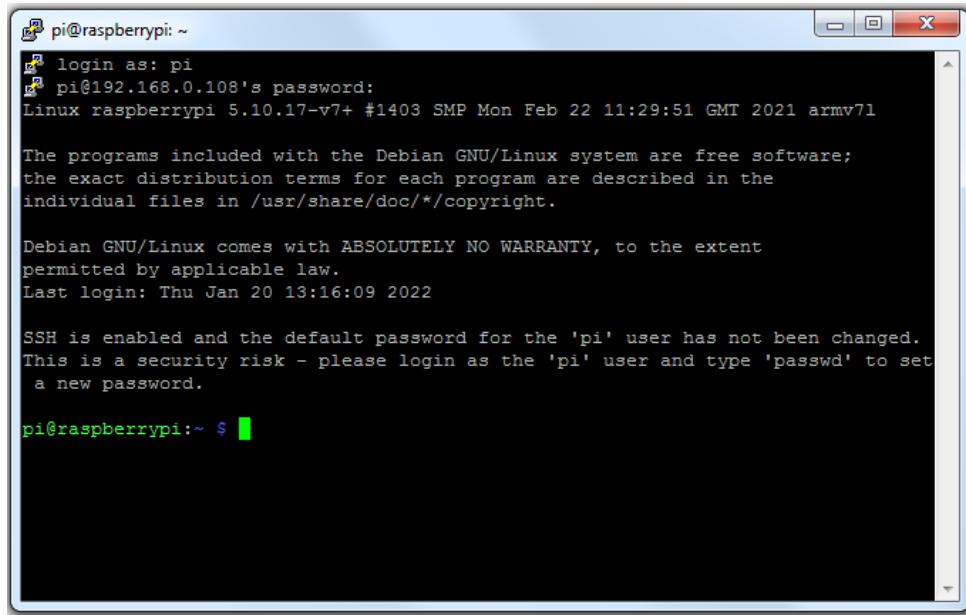
```

Reconocerá la placa Raspberry Pi por su dirección MAC, pues contiene el nombre de la empresa que fabrica el dispositivo, en este caso Raspberry Pi Foundation.

- III. Acceder a la terminal de comandos: se recomienda descargar el programa Putty, el cual le permitirá acceder de forma remota a la terminal del dispositivo con su IP y su puerto. Solicitará un usuario y contraseña que por defecto en estas placas el usuario es 'pi' y la contraseña 'raspberry'.

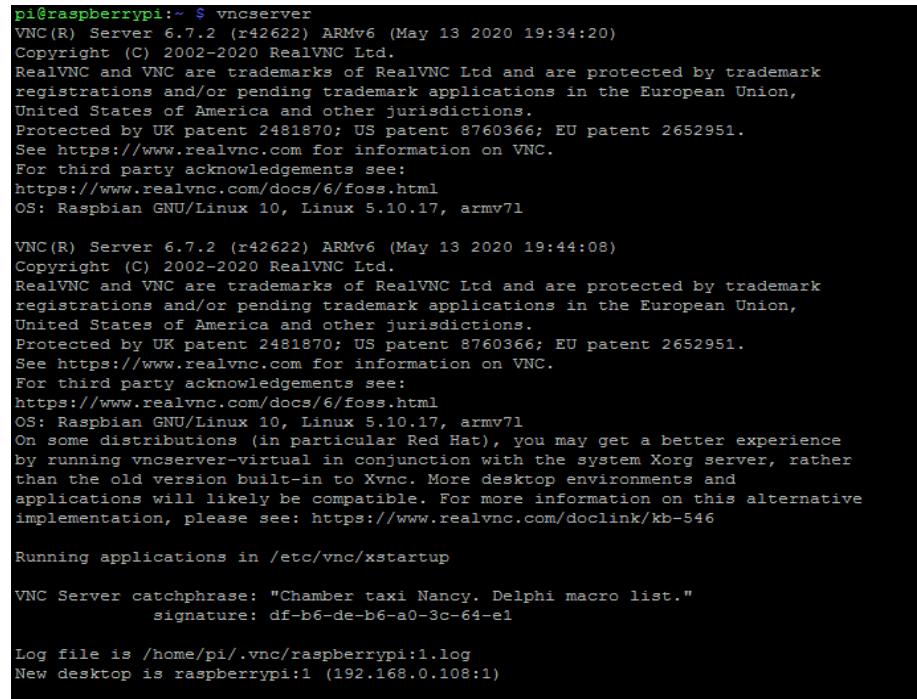


Deberá aparecer la línea de comandos del dispositivo:



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the standard Raspbian login screen with the following text:
login as: pi
pi@192.168.0.108's password:
Linux raspberrypi 5.10.17-v7+ #1403 SMP Mon Feb 22 11:29:51 GMT 2021 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 20 13:16:09 2022
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.
pi@raspberrypi:~ \$

- IV. Si desea acceder a la interfaz gráfica deberá instalar otro programa, tanto en su computadora como en la Raspberry Pi. Desde la terminal de la placa ejecute el comando ***sudo apt-get install –y realvnc-vnc-server realvnc-vnc-viewer***. En caso de que el comando anterior falle, instale primero vnc server con ***sudo apt-get install realvnc-vnc-server*** y luego vnc viewer con ***sudo apt-get install realvnc-vnc-viewer***. Una vez instalado ejecute ***vncserver*** lo que le permitirá ejecutar un servidor vnc.



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the output of the "vncserver" command:
pi@raspberrypi:~ \$ vncserver
VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:34:20)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See <https://www.realvnc.com> for information on VNC.
For third party acknowledgements see:
<https://www.realvnc.com/docs/6/foss.html>
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l

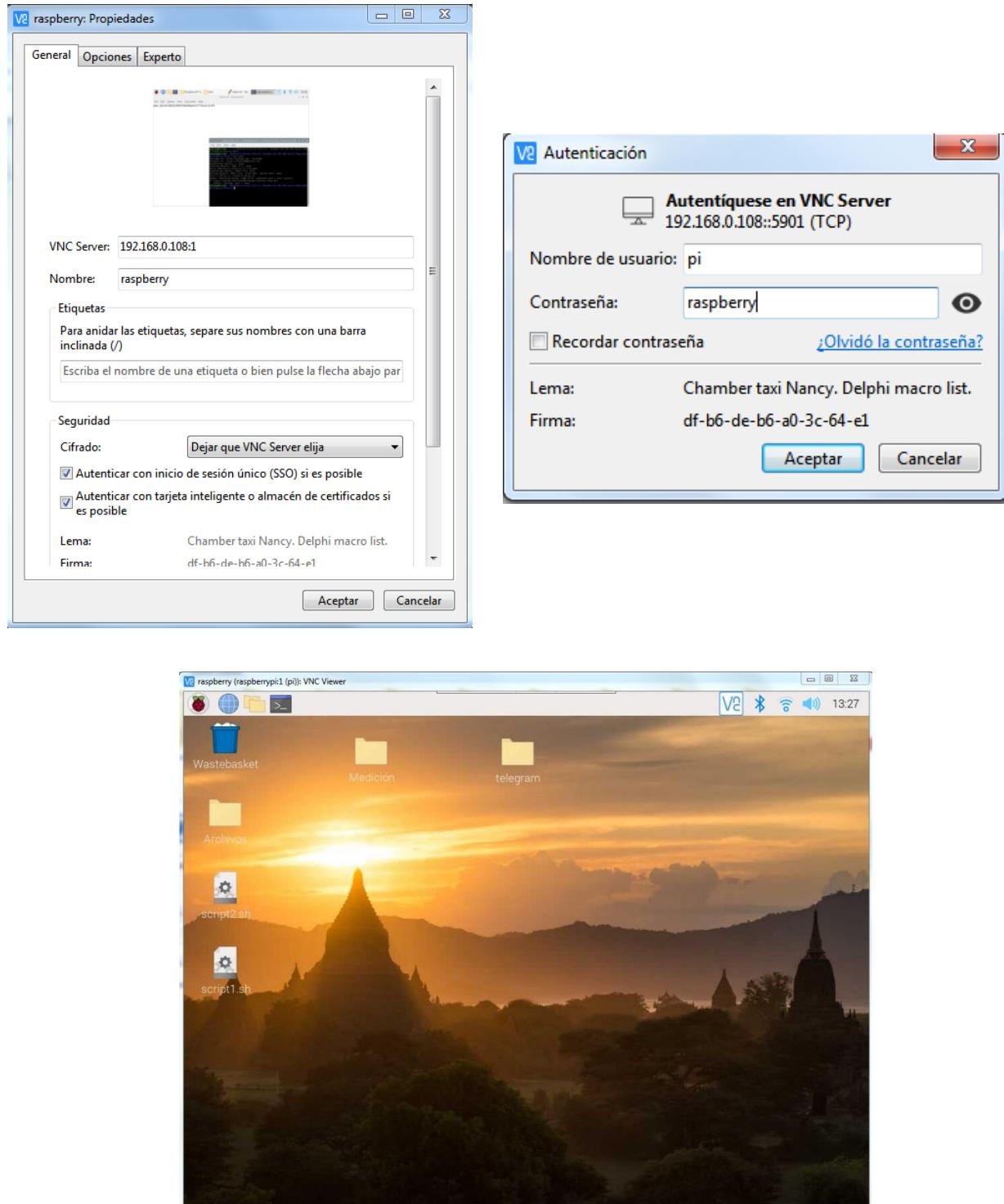
VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:44:08)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See <https://www.realvnc.com> for information on VNC.
For third party acknowledgements see:
<https://www.realvnc.com/docs/6/foss.html>
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l
On some distributions (in particular Red Hat), you may get a better experience
by running vncserver-virtual in conjunction with the system Xorg server, rather
than the old version built-in to Xvnc. More desktop environments and
applications will likely be compatible. For more information on this alternative
implementation, please see: <https://www.realvnc.com/doclink/kb-546>

Running applications in /etc/vnc/xstartup

VNC Server catchphrase: "Chamber taxi Nancy. Delphi macro list."
signature: df-b6-de-b6-a0-3c-64-e1

Log file is /home/pi/.vnc/raspberrypi:1.log
New desktop is raspberrypi:1 (192.168.0.108:1)

Generará un nuevo escritorio con la dirección IP asignada a la placa seguido de dos puntos y un número. Descargue el programa VNC Viewer en su computadora y una vez instalado coloque el escritorio generado por el servidor; también pedirá un identificador y luego el usuario y contraseña que son las ya mencionadas anteriormente. De esta forma podrá acceder al escritorio de la placa Raspberry Pi como si la hubiera conectado de forma directa a un monitor, teclado y mouse.



- **Paso 4: Instalación de librerías y compilador.**

- **Programa de medición.**

Las librerías y funciones empleadas en el programa encargado de realizar las mediciones fueron obtenidas a través de la página <https://github.com/AKEOPLUS-boris-bocquet/RaspberryPi-ADC-DAC>. En la misma se descargan las librerías y códigos para realizar la conversión analógica a digital (ADC, usando el circuito ADS1256) y digital a analógico (DAC, usando DAC8552) basados en Raspberry PI extendido con Waveshare High-Precision AD-DA Raspberry Pi hat/shield (placa conversora).

El primer paso es descargar e instalar el compilador *cmake*, así como también el software de control de versiones *Git* con el fin de descargar estas librerías desde *GitHub*.

Aplicar los comandos:

```
sudo apt-get update  
sudo apt-get install git cmake.
```

En caso de que no se instalen todas las dependencias, pruebe con `sudo apt --fix-broken install` y vuelva a probar `sudo apt-get install git cmake`.

El segundo paso es crear el directorio *Archivos* en el escritorio. Dentro de esta carpeta crear el directorio *librería*.

Deberá descargar la biblioteca BCM2835 y sus encabezados. Descargar la última versión de la biblioteca `bcm2835-1.xx.tar.gz` directamente desde la página y colocarla en el directorio *Archivos*.

Luego ejecute los siguientes comandos (reemplace `xx` en el primer y segundo comando por el número correspondiente a la versión de la biblioteca descargada):

```
tar zxvf bcm2835-1.xx.tar.gz  
cd bcm2835-1.xx  
.configure  
make  
sudo make check  
sudo make install
```

Luego puede clonarse el repositorio a la placa Raspberry Pi dentro del directorio *librería*:

```
git clone https://github.com/AKEOPLUS-boris-bocquet/RaspberryPi-ADC-DAC.git  
cd RaspberryPi-ADC-DAC  
mkdir build  
cd build  
cmake ..  
make
```

Lo siguiente es habilitar la comunicación SPI ya que por defecto viene deshabilitada. Existen dos formas de habilitarla:

1. Desde la terminal:

Aplicar `sudo raspi-config` con lo cual accederá a las configuraciones de la Raspberry Pi y luego ir a *5 – Interfacing Options* y *P4 - SPI* y colocar <Yes> para habilitar SPI.

2. Desde el GUI de la Raspberry Pi dirigirse a las configuraciones, luego a opciones de interfaz y entonces marcar el casillero de habilitar en la comunicación SPI.

Si se ha instalado todo correctamente, compile el programa *mainTestingAdda.c* con `sudo make mainTestingAdda.c`. Deberá crearse el archivo ejecutable *testAdda*.

A modo de prueba, con el fin de determinar si la placa se encuentra en funcionamiento y puede ejecutarse correctamente el programa de prueba, ejecute el programa *mainTestingAdda.c*, que es el ejemplo de prueba:

```
sudo ./testAdda
```

Se mostrarán por consola las sucesivas mediciones que se realizan en cada uno de los 8 canales del ADC.

Consideraciones:

- Notar que la ejecución de los programas se realiza con `sudo`, ya que se requiere permisos de root o super usuario para poder ejecutarlas.
- Si clona el repositorio en el directorio librería, el mismo tendrá el nombre ‘RaspberryPi-ADC-DAC’. Sin embargo, si descarga la carpeta comprimida y la descomprime en el directorio librería, el nombre de la carpeta será ‘RaspberryPi-ADC-DAC-master’. Esto no afecta al normal funcionamiento del programa de medición, pero si al programa de acceso remoto de Telegram , ya que este último utiliza la ruta completa donde se encuentran los archivos generados. Por esto, si decide descargar y descomprimir, deberá modificar el programa de acceso remoto y colocar “-master” en todas las líneas que hagan referencia a la ruta de búsqueda de archivos.
- Por defecto con el sistema operativo viene instalado el editor de texto Geany. Con el mismo podrá modificar los programas en C y compilarlos.

- **Programa de acceso remoto por Telegram.**

Para lograr la comunicación remota entre el usuario y la Raspberry Pi, se hizo uso de la librería y las funciones obtenidas desde GitHub a través del siguiente link:

<https://github.com/smartnode/telebot>

Para la instalación de las librerías, en las distribuciones de Linux basadas en Debian, puede hacerlo de la siguiente manera:

```
sudo apt-get install libcurl4-openssl-dev libjson-c-dev cmake binutils make
```

Cree el directorio *telegrambot* dentro de la carpeta *Archivos*. En este directorio descargue y descomprima el archivo comprimido desde la página mencionada.

Dentro de la carpeta *telebot-master* ingrese al directorio *test*. Para construir la biblioteca, ejecute los siguientes comandos:

```
cmake ../  
make
```

El programa base en el directorio *test* es *echobot.c* y al compilar con *make* se crea el ejecutable *echobot*.

Atención: si intenta ejecutar este programa sin haber realizado los pasos que se detallan a continuación, probablemente fallará la ejecución debido a que requiere una clave *token* de un bot que todavía no se ha creado.

- **Paso 5: Programas principales.**

Copie el programa de medición *vibration_measurement.c* y péguelo en el directorio *home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC/*.

Para compilar este programa modifique el archivo *CMakeList.txt* en la misma carpeta:

```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)  
project(TestingAdda)  
SET(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall")  
if(WIN32)  
else()  
    SET(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wextra -Werror")  
endif()  
  
SET(MOCK_SPI False CACHE BOOL "On : SPI communication is mocked. Off : use real SPI lib such as BCM2835  
SET(USE_BCM True CACHE BOOL "On : use BCM2835 as SPI lib")  
  
add_subdirectory(AD-DA-WS-RPI)  
  
add_executable(testAdda vibration_measurement.c)  
target_link_libraries(testAdda ADDA_WS_RPI)
```

Coloque el nombre del programa en c y el nombre del ejecutable en las líneas recuadradas.

Consideración: los nombres mencionados del programa y el archivo ejecutable no son taxativos. Sin embargo, si desea modificarlos debe tenerlos en cuenta en la creación de los servicios para su arranque automático.

Si ya ha compilado anteriormente el programa de prueba *mainTestingAdda.c*, elimine la carpeta *build* antes de compilar el nuevo programa. Caso contrario, pueden ocurrir fallos de compilación.

Creada nuevamente la carpeta *build*, ingrese a esta y compile:

```
cd build
```

```
cmake ..  
make
```

Luego copie y reemplace el programa *echobot.c* en el directorio *home/pi/Desktop/Archivos/telegrambot/telebot-master/test*. También puede copiar el código y reemplazar el existente en el programa que ya se encuentra en dicha carpeta.

Debido a que no se ha modificado el nombre del programa, podrá compilarlo sin modificar ningún otro archivo. Sin embargo, se recomienda modificar cualquier línea del nuevo programa y nuevamente compilar para generar correctamente el nuevo ejecutable. En caso de que desee cambiar el nombre del porgrama, modifique el archivo *CMakeList.txt* en la misma carpeta:

```
SET(TEST_NAME echobot)  
SET(TEST_SRC echobot.c)  
ADD_EXECUTABLE(${TEST_NAME} ${TEST_SRC})  
TARGET_LINK_LIBRARIES(${TEST_NAME} ${PKGS_LDFLAGS} ${PROJECT_NAME} pthread)  
  
#EOF
```

No ejecute el programa hasta no haber creado el bot.

- **Paso 6: Archivos adjuntos.**

Para el correcto funcionamiento de los programas, se requieren algunos archivos que deben ubicarse en las carpetas que contienen los ejecutables de los programas.

- **Archivos adjuntos al programa de medición.**
 1. Archivo “Parametros.txt”: este archivo contiene los parámetros de configuración que utilizará el programa de medición.
 2. Archivo “Parametros_respaldo.txt”: es un archivo de respaldo para el archivo “Parametros.txt”.
 3. Archivo “lastfile.txt”: este archivo contendrá el número del siguiente archivo a crear. Este archivo puede no copiarlo ya que, en caso de no existir, el mismo programa se encargará de crearlo.

Copiar archivos en el directorio *build* en la ruta donde se encuentra el ejecutable:

home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC/build

Si el programa *vibration_measurement.c* compiló correctamente ya está en condiciones de ejecutarlo:

```
sudo ./testAdda
```

- **Archivos adjuntos al programa de acceso remoto por Telegram.**

1. Archivo “menu.txt”: es un archivo de texto que contiene el menú principal que será presentado al usuario.
2. Archivo “token.txt”: este archivo contendrá el token del bot.

3. Archivo “resultado.txt”: archivo de texto donde se guardará el nombre y la ruta de los archivos buscados.
4. Archivo “temperatura.txt”: archivo auxiliar donde se guardará la temperatura medida de la placa.
5. Imagen “download.jpeg”: imagen de muestra. No afecta al normal funcionamiento del programa.

Copiar archivos en el directorio *test* en la ruta donde se encuentra el ejecutable:

home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test

- **Paso 7: Creación del bot de Telegram.**

En primer lugar, debe tener Telegram instalado con una cuenta habilitada. Lo siguiente es ir a los contactos de Telegram y buscar:

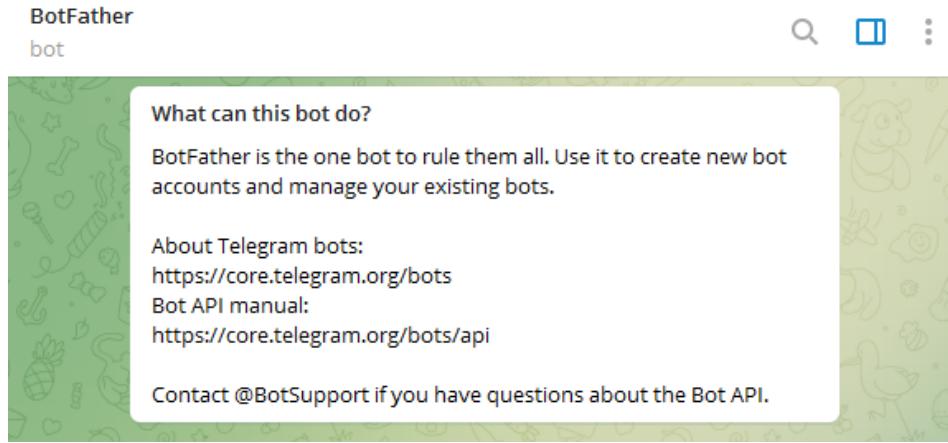
@BotFather

Le aparecerá algo como lo siguiente:

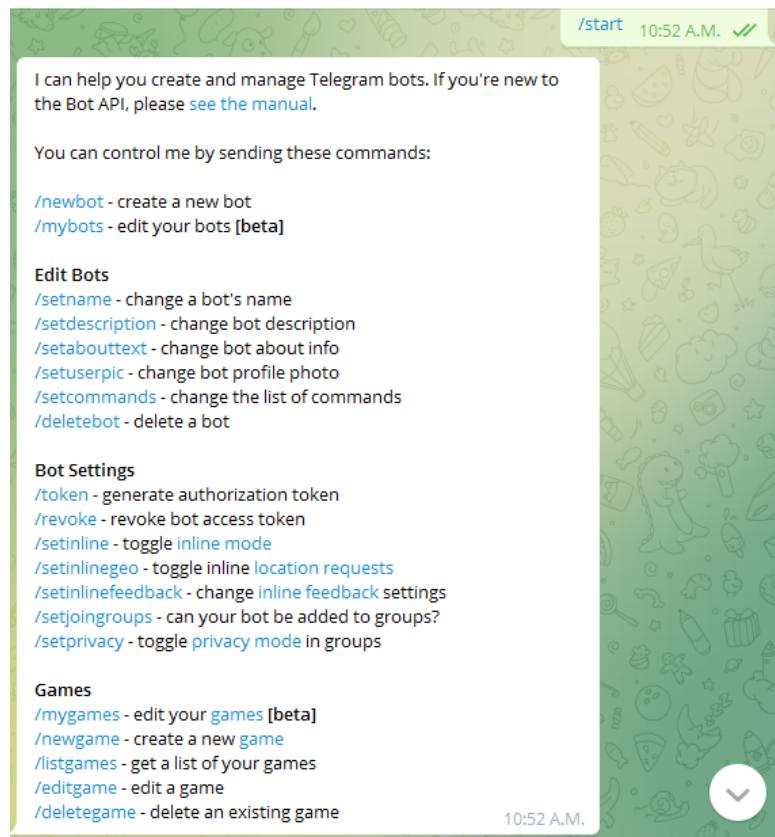


A pesar de que aparecen varias opciones sólo la primera es la que utilizará. Note que posee una tilde azul para diferenciarlo de los demás. Este BotFather no es más que otro bot dedicado exclusivamente a la creación de nuevos bots.

Lo primero que este bot muestra es el siguiente mensaje:



Colocando el comando **/start** mostrará un listado de opciones con las acciones que este bot puede realizar:

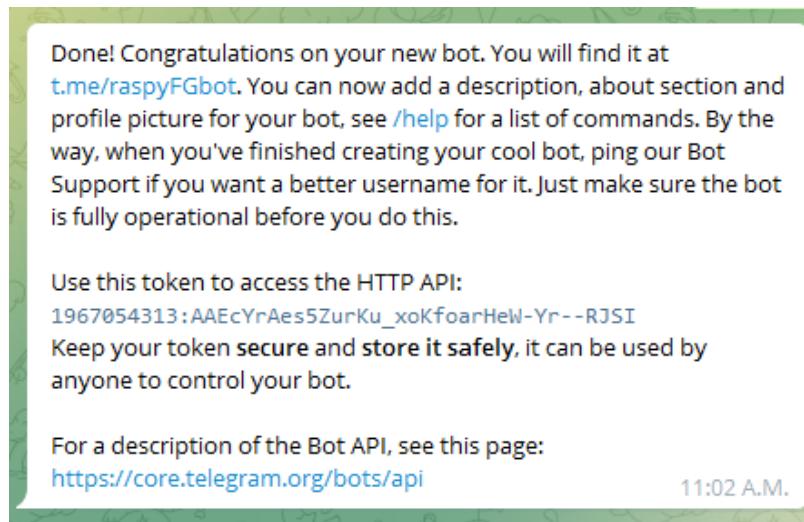


Con el comando **/newbot** podrá crear un nuevo bot:



Se debe ingresar un nombre para el bot y un nombre de usuario al mismo.

Una vez realizados estos pasos, ya se ha creado el bot, el cual posee un 'token', que es como una clave de identificación del bot.



La primera prueba inicial que puede realizar para saber si su bot se ha creado correctamente es enviarle un mensaje desde el navegador. Para ello utilice:

[https://api.telegram.org/bot\[TOKEN\]/sendMessage?chat_id=\[CHAT_ID\]&text\[MESSAGE\]](https://api.telegram.org/bot[TOKEN]/sendMessage?chat_id=[CHAT_ID]&text[MESSAGE])

Por ejemplo:

https://api.telegram.org/bot1967054313:AAEcYrAes5ZurKu_xoKfoarHeW-Yr--RJSI/sendMessage?chat_id=1980654556&text=hola mundo

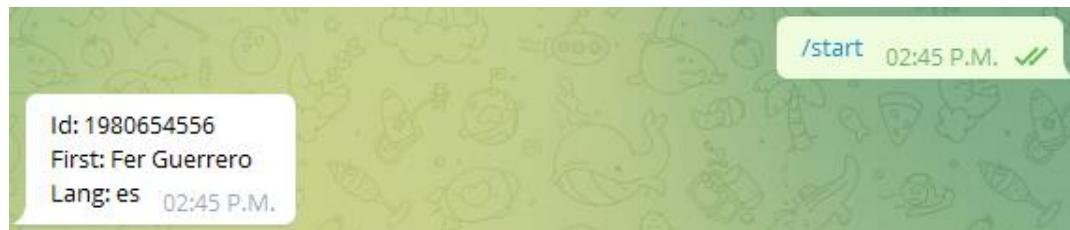
De haberse enviado correctamente, debería aparecer el siguiente mensaje:

```
{"ok":true,"result":{"message_id":6311,"from":{"id":1967054313,"is_bot":true,"first_name":"raspybot","username":"raspyFGbot"},"chat":{"id":1980654556,"first_name":"Fer Guerrero","type":"private"},"date":1643133576,"text":"hola mundo"}}
```

Y aparecerá en el chat del bot el mensaje enviado:



Para obtener la identificación de su chat puede hacerlo a través del bot **@userinfobot**, el cual al iniciararlo arrojará el chat id:

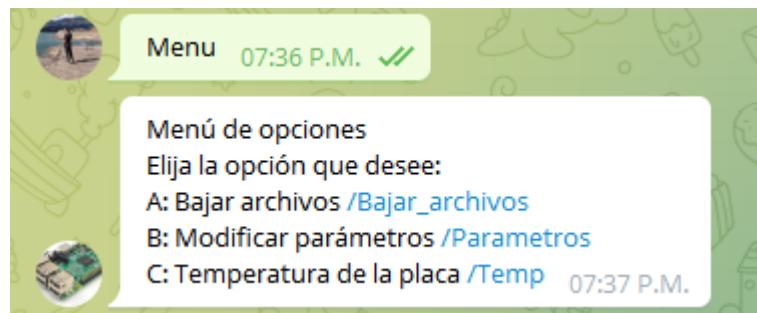


Ya creado el bot, copie el token en el archivo "token.txt" en el directorio
home/pi/Desktop/Archivos/telegrambot/telebot-master/test/test.

Ahora está en condiciones de ejecutar el programa de acceso remoto.

```
sudo ./echobot
```

Envíe el mensaje **Menu** (sin acento) a través de su chat al bot para verificar su funcionamiento. Debería presentarle el menú principal de opciones:



- **Paso 8: Instalación de módulo RTC.**

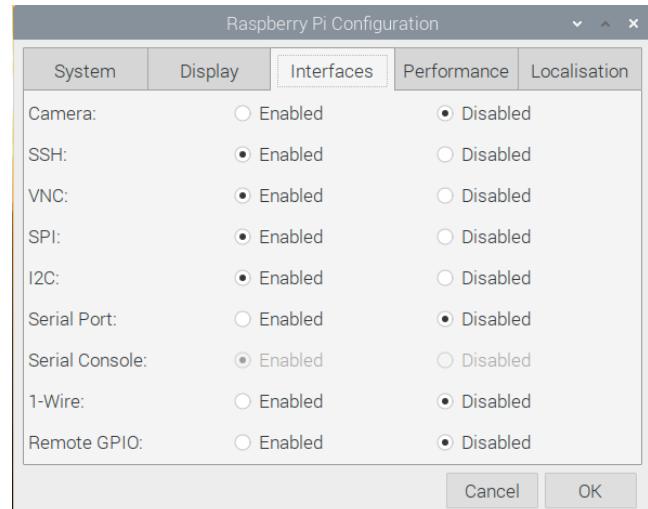
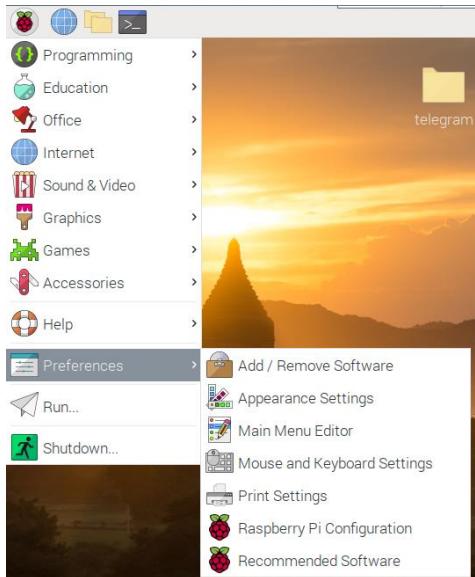
1. Actualizar:

Antes de empezar la instalación, debe asegurarse de que todos los paquetes estén actualizados, por lo que, primero se debe realizar una actualización y luego instalar el software I2C:

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install i2c-tools
```

```
pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get upgrade - yes
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.7 kB]
Get:3 http://archive.raspberrypi.org/debian buster/main armhf Packages [393 kB]
Reading package lists... Done
E: Repository 'http://raspbian.raspberrypi.org/raspbian buster InRelease' changed its 'Suite' value from 'stable' to 'oldstable'
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.
pi@raspberrypi:~ $ sudo apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (4.1-1).
i2c-tools set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 113 not upgraded.
```

También debe activar el bus I2C desde la configuración de la Raspberry Pi:



2. Detectar:

Lo siguiente es determinar si la placa detecta al módulo. Para eso utilice el comando:

```
sudo i2cdetect -y 1
```

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
Error: Could not open file '/dev/i2c-1' or '/dev/i2c/1': No such file or directory
pi@raspberrypi:~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: ----- -
10: ----- -
20: ----- -
30: ----- -
40: ----- -
50: ----- -
60: ----- 68 -----
70: ----- -
```

Si el resultado es el ID #68 significa que el módulo RTC funciona ya que es la dirección normalmente asignada al RTC DS3231 y también del DS1307.

Se puede agregar un soporte para el RTC agregando una superposición del árbol de dispositivos. Para ello debemos editar el archivo config.txt.

```
sudo nano /boot/config.txt
```

Para editar la configuración de la placa y agregar aquel que coincide con su chip RTC, escriba:

```
dtoverlay=i2c-rtc,ds1307
```

o

```
dtoverlay=i2c-rtc,ds3231
```

al final del archivo.

```
GNU nano 3.2                               /boot/config.txt

dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d

# NOOBS Auto-generated Settings:

 dtoverlay=i2c-rtc,ds3231
```

Guarde y ejecute `sudo reboot` para reiniciar la placa. Luego ejecute `sudo i2cdetect -y 1`, deberá verse UU donde antes estaba la dirección 0x68.

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: -----
20: -----
30: -----
40: -----
50: -----
60: -- UU --
70: -----
```

3. Cargar el módulo en el kernel:

Antes de avanzar es importante aclarar que el módulo RTC DS3231 y el DS1307 son compatibles y sustitutos, por lo tanto, se puede instalar el DS3231 con los mismos comandos del DS1307 e incluso colocando el nombre ds1307 en la sintaxis usada.

En base a lo dicho, ejecute `sudo modprobe rtc-ds1307`.

En súper usuario: `sudo bash`

Y luego:

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
o
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Luego salga del super usuario:

```
exit
```

Puede comprobar los módulos cargados en el kernel en la ruta `/lib/modules/5.10.17-v7+` (este último directorio puede variar de nombre) y abriendo el archivo `modules.alias` verá los alias de los distintos módulos cargados:

```

modules.alias - Mousepad
File Edit Search View Document Help
alias i2c:m41t11 rtc_ds1307
alias i2c:m41t00 rtc_ds1307
alias i2c:m41t0 rtc_ds1307
alias i2c:ds3231 rtc_ds1307
alias i2c:ds1341 rtc_ds1307
alias i2c:ds1340 rtc_ds1307
alias i2c:ds1388 rtc_ds1307
alias i2c:ds1339 rtc_ds1307
alias i2c:ds1338 rtc_ds1307
alias i2c:ds1337 rtc_ds1307
alias i2c:ds1308 rtc_ds1307
alias i2c:ds1307 rtc_ds1307
alias of:N*T*Cepson,rx8130C* rtc_ds1307
alias of:N*T*Cepson,rx8130 rtc_ds1307
alias of:N*T*Cisil,isl12057C* rtc_ds1307
alias of:N*T*Cisil,isl12057 rtc_ds1307
alias of:N*T*Cepson,rx8025C* rtc_ds1307
alias of:N*T*Cepson,rx8025 rtc_ds1307
alias of:N*T*Cpericom,pt7c4338C* rtc_ds1307
alias of:N*T*Cpericom,pt7c4338 rtc_ds1307
alias of:N*T*Cmicrochip,mcp7941xC* rtc_ds1307
alias of:N*T*Cmicrochip,mcp7941x rtc_ds1307
alias of:N*T*Cmicrochip,mcp7940xC* rtc_ds1307
alias of:N*T*Cmicrochip,mcp7940x rtc_ds1307
alias of:N*T*Cst,m41t11C* rtc_ds1307
alias of:N*T*Cst,m41t11 rtc_ds1307
alias of:N*T*Cst,m41t00C* rtc_ds1307
alias of:N*T*Cst,m41t00 rtc_ds1307
alias of:N*T*Cst,m41t0C* rtc_ds1307
alias of:N*T*Cst,m41t0 rtc_ds1307
alias of:N*T*Cmaxim,ds3231C* rtc_ds1307
alias of:N*T*Cmaxim,ds3231 rtc_ds1307
alias of:N*T*Cdallas,ds1341C* rtc_ds1307

```

Ya se encuentra cargado el módulo al kernel y cada vez que la Raspberry se inicie podrá comprobar la hora en el RTC:

```
sudo hwclock -r
```

```

pi@raspberrypi:~ $ sudo hwclock -r
2021-09-21 20:17:32.807871-03:00

```

Si es la primera vez que lo ejecuta y no posee conectividad a internet la Raspberry Pi devolverá la fecha 1 de enero de 2000. Para configurar la hora, la manera más sencilla es conectarla a internet para que automáticamente se actualice.

Si no, de forma manual, primero deberá configurar la hora del sistema y luego escribirla en el RTC. Esto se puede hacer con:

```
sudo hwclock --set --date = "$(date" +%m /%d /%Y %H:%M:%S ")"
```

o con:

```
sudo date 08050822.
```

Donde:

08 es el mes
05 es el día
08 es la hora
22 son los minutos

Y luego sudo hwclock -w.

4. Lograr que la Raspberry obtenga la hora del RTC:

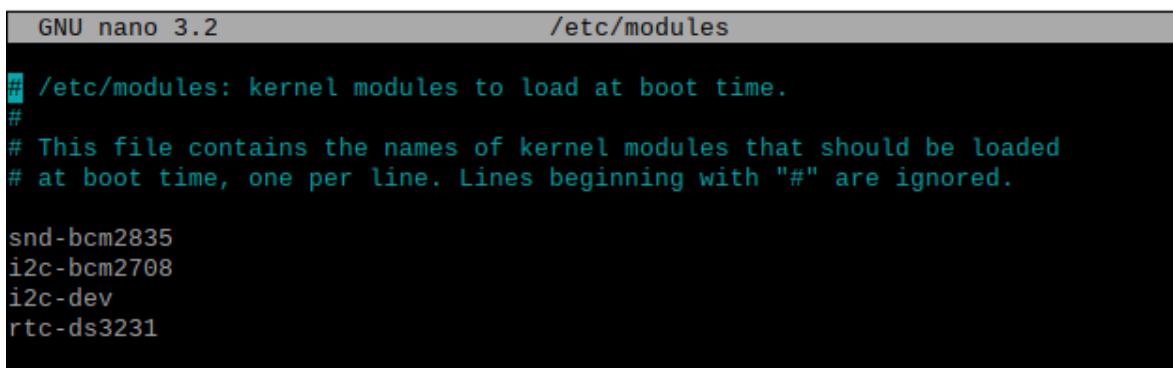
Ya se ha conseguido que la Raspberry Pi configure la hora en el RTC, sin embargo, se le debe indicar que obtenga la hora del RTC cada vez que se inicie. Para ello hay que modificar el archivo *modules* en el directorio *etc*.

Este archivo contiene los nombres de los módulos del kernel que se deben cargar en el momento del arranque.

```
sudo nano /etc/modules
```

En el mismo se deben agregar las siguientes líneas al final del archivo:

```
snd-bcm2835  
i2c-bcm2708  
i2c-dev  
rtc-ds3231
```



```
GNU nano 3.2          /etc/modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

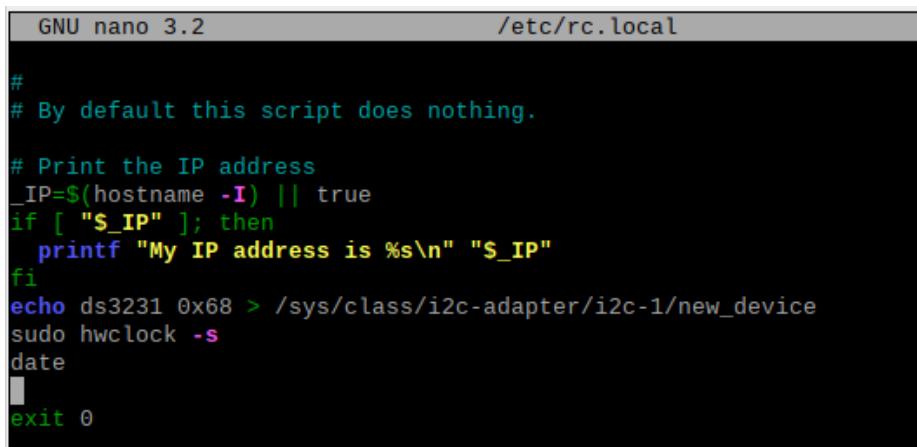
snd-bcm2835
i2c-bcm2708
i2c-dev
rtc-ds3231
```

El siguiente paso es añadir el dispositivo DS3231 como reloj del sistema en el archivo *rc.local*. Para ello ejecutar:

```
sudo nano /etc/rc.local
```

Y añadir las siguientes líneas justo antes de «*exit 0*»

```
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sudo hwclock -s
date
```



```
GNU nano 3.2          /etc/rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi
echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sudo hwclock -s
date

exit 0
```

Guarde el archivo, desconecte la conexión a internet y reinicie la Raspberry Pi para probar el correcto funcionamiento del módulo.

- **Paso 9: Creación de servicios y mensajes de inicio.**

Para que los programas se inicien de forma automática en el arranque del sistema operativo se deben crear dos servicios encargados de esto.

- **Servicio de medición:**

Para crear el servicio de medición en *systemd*, ejecute:

```
sudo nano /etc/systemd/system/raspy.service
```

Dentro de este archivo coloque y guarde:

```
[Unit]
Description= Servicio de Medición
After= multi-user.target

[Service]
Type=simple
Restart=always
RestartSec=3
ExecStart= sudo
./home/pi/Desktop/Archivos/libreria/RaspberryPi-ADC-DAC
/build/testAdda
#ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

- **Servicio de Telegram:**

Para el programa de comunicación remota con Telegram cree el servicio *telegram.service* con:

```
sudo nano /etc/systemd/system/telegram.service
```

Dentro de este archivo copie y guarde las siguientes instrucciones:

```
[Unit]
Description= Servicio Telegram
Requires=network.target
After=network.target

[Service]
Type=simple
```

```

Restart=always
RestartSec=3
ExecStart= sudo
./home/pi/Desktop/Archivos/telegrambot/telebot-
master/test/test/echobot
#ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target

```

- **Inicio del servicio:**

Configurar los permisos del archivo:

```
sudo chmod 755 /etc/systemd/system/raspy.service
```

Cargar el servicio al sistema init:

```
sudo systemctl daemon-reload
```

Iniciar el servicio:

```
sudo systemctl start raspy.service
```

Habilitar el servicio para que se ejecute automáticamente en el arranque del sistema:

```
sudo systemctl enable raspy.service
```

Con estos pasos el servicio ya se encuentra en funcionamiento. Si desea ver el estado del servicio ejecute:

```
sudo systemctl status raspy.service
```

```

● raspy.service - Servicio Medición
   Loaded: loaded (/etc/systemd/system/raspy.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-02-04 11:41:20 -03; 40s ago
     Main PID: 789 (sh)
        Tasks: 3 (limit: 2062)
       CGroup: /system.slice/raspy.service
               └─789 /usr/bin/sh /home/pi/Desktop/script1.sh
                 ├─794 sudo ./testAdda
                 ├─796 ./testAdda
                 ~

Feb 04 11:41:20 raspberrypi systemd[1]: Started Servicio Medición.
Feb 04 11:41:20 raspberrypi sudo[794]:      pi : TTY=unknown ; PWD=/home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build ; USER=root ; COMMAND=./testAdda
Feb 04 11:41:20 raspberrypi sudo[794]: pam_unix(sudo:session): session opened for user root by (uid=0)
~
~
~
```

Donde puede observarse que se encuentra en estado activo y ejecutándose.

Para el servicio de Telegram realice los mismos pasos colocando telegram.service:

Si desea detener el servicio:

```
sudo systemctl stop raspy.service
```

O reiniciarlo:

```
sudo systemctl restart raspy.service
```

Importante: si desea ejecutar los programas desde la terminal de comandos en forma manual, primero debe detener estos servicios. Caso contrario, el programa de medición fallará debido a que ya existe otro proceso utilizando el ADC y el programa de comunicación remota enviará los mismos mensajes y descargará los mismos archivos dos veces debido a que se ejecuta simultáneamente dos veces.

▪ Avisos de inicio

Por último, se debe configurar la placa para que al iniciarse esta o establecer alguna conexión SSH, la misma envíe un mensaje al bot indicando la situación.

Para esto se crean dos scripts. El primero es el script *prueba.sh* encargado de enviar un mensaje cuando la placa se inicia:

```
#!/bin/bash
TOKEN="Coloque aquí el token del bot"
ID="Coloque aquí el ID de su chat"
MENSAJE="Se reinició la placa."
URL="https://api.telegram.org/bot$TOKEN/sendMessage?chat_id=$ID
curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE"
```

Genere este script en */home/pi/Desktop*.

Si ejecuta este script se enviará el mensaje “Se reinició la placa” al bot cuyo token se ha colocado. Para que dicho mensaje se envíe automáticamente al inicio de la placa deberá añadir este script a las tareas del Cron de Linux. Estas tareas se ejecutan de forma automática al inicio de la placa.

Para ello escriba el siguiente comando en la terminal:

```
sudo crontab -e
```

Y agregue al final del archivo la siguiente línea:

```
@reboot ( sleep 30 ; sh /home/pi/Desktop/prueba.sh )
>/dev/null 2>&1
```

```

# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
@reboot ( sleep 30 ; sh /home/pi/Desktop/prueba.sh ) >/dev/null 2>&1

```

Guarde el archivo y reinicie la placa. Deberá aparecer el siguiente mensaje en el chat transcurrido 30 segundos:



En caso de que tenga problemas con el formato del string a enviar, pruebe modificar el archivo a formato UTF-8 o cambiar el mensaje a “Se ha reiniciado la placa.”.

El segundo script, llamado *prueba2.sh* envía un mensaje cada vez que un usuario se conecta a la placa mediante una conexión SSH. También indica la dirección IP del usuario conectado.

```

#!/bin/bash

TOKEN="Coloque aquí su token"
ID="Coloque aquí el ID de su chat"
MENSAJE="Alguien se acaba de conectar por SSH con la IP $(echo
$SSH_CLIENT | awk '{print $1}')."
URL="https://api.telegram.org/bot$TOKEN/sendMessage?chat_id=$I
D=$MENSAJE"

curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE" -d
parse_mode=$HTML

```

Genere este script en */home/pi/Desktop*.

Cada vez que se inicia una sesión en el terminal, el sistema lee y ejecuta el archivo llamado «.bashrc». Debe editar este archivo. Ejecute los siguientes comandos:

```

cd /home/pi
sudo nano .bashrc

```

Y agregue la siguiente línea al final del archivo:

```
sh /home/pi/Desktop/prueba2.sh >/dev/null 2>&1
```

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi
sh /home/pi/Desktop/prueba2.sh >/dev/null 2>&1
```

De esta forma, cada vez que un usuario se conecte a través de una conexión SSH, por ejemplo, al abrir la terminal de comandos, se enviará este mensaje:

Alguien se acaba de conectar por SSH con la IP 192.168.0.110.

12:06 P.M.

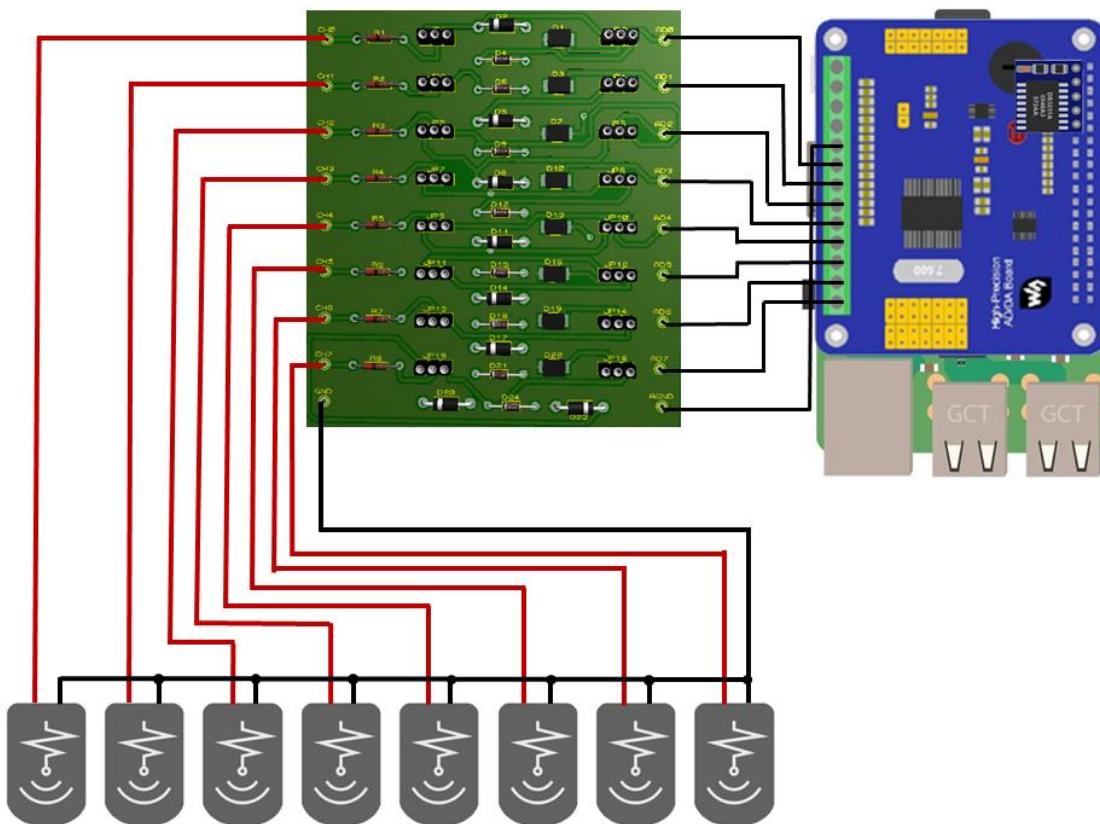
Reinicie la placa y compruebe que lleguen los mensajes de inicio.

11 ANEXO: Manual de Usuario

El siguiente manual pretende guiar al usuario en el proceso de medición, obtención de las muestras y acceso remoto al dispositivo.

11.1 Conexionado

La conexión de los sensores de medición de vibraciones debe realizarse a las entradas del circuito de protección.



Se recomienda realizar las conexiones de los sensores al dispositivo con este último sin alimentación para evitar cualquier posible falso contacto que podría dañar al dispositivo o a los sensores.

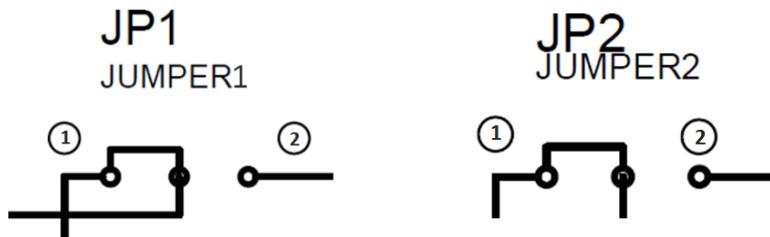
A través de dos jumpers en el circuito de protección, uno a la entrada y uno a la salida, puede elegir el rango de entrada para cada canal de entrada, para el caso en que el buffer se encuentre habilitado o que se encuentre deshabilitado.

Una tabla de verdad con las 4 posiciones posibles de los jumpers se muestra a continuación:

Posición	Jumper 1	Jumper 2	Límite superior	Límite inferior
Posición 1	SI	SI	5.1 V	-0.19 V
Posición 2	NO	NO		
Posición 1	SI	NO	—	—

Posición 2	NO	SI		
Posición 1	NO	SI	—	—
Posición 2	SI	NO	—	—
Posición 1	NO	NO	3.3 V	-0.19 V
Posición 2	SI	SI		

Donde:



Nota: aunque el rango de tensión de entrada puede seleccionarse en forma individual para cada entrada, la habilitación o des habilitación del buffer analógico de entrada afecta a todas las entradas analógicas en forma simultánea.

11.2 Encendido y acceso al dispositivo

La placa Raspberry Pi 3 B+, toma su alimentación a través de un adaptador 220VAC/5VDC conectado a la red eléctrica. El mismo viene incluido con la placa. Al conectarla el dispositivo se encenderá automáticamente.

Si se desea tener conexión a internet, esta puede obtenerse a través de un cable ethernet conectado de forma directa a la Raspberry Pi o a través de conexión Wi-Fi una vez que el sistema operativo haya iniciado.

El dispositivo se encuentra configurado para comenzar a medir automáticamente una vez iniciado el sistema operativo.

Para acceder al dispositivo puede hacerlo de dos formas:

- **Acceso directo:**

Puede acceder al escritorio del dispositivo conectando la placa Raspberry Pi a un mouse, teclado y monitor. Esta placa posee 4 puertos USB 2.0, un puerto HDMI y un puerto Ethernet para conexión a internet. Adicionalmente posee un puerto CSI para conectar una cámara y un puerto DSI para conectar una pantalla táctil.

- **Acceso remoto:**

Se hará una división entre el acceso remoto dentro de la misma red de área local y otra fuera de esta.

➤ **Dentro de la red de área local.**

Se requiere conocer la dirección IP y el puerto de la placa Raspberry Pi antes de realizar cualquier conexión.

Para ello se recomienda el programa Nmap, el cual escanea todas las direcciones IP dentro de la red y muestra un listado de aquellas que se encuentran ocupadas. Para esto se ejecuta nmap desde la terminal de comandos a través del comando **XXX.XXX.XXX/MM** colocando el nombre de su red con su respectiva máscara. Con esto se le indica desde cual dirección IP dentro de la red desea que escanee. Cuando finalice arrojará un listado de las IP ocupadas e información relacionada como por ejemplo el puerto que está utilizando y la dirección MAC de cada dispositivo.

Reconocerá la placa Raspberry Pi por su dirección MAC, pues contiene el nombre de la empresa que fabrica el dispositivo, en este caso Raspberry Pi Foundation.

```
C:\Users\Administrador>nmap 192.168.0.0/24
Starting Nmap 7.91 < https://nmap.org > at 2022-01-20 13:17 Hora estàndar de Argentina
Nmap scan report for 192.168.0.1
Host is up <0.023s latency>.
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 58:D9:D5:28:C7:18 (Tenda Technology,Ltd.Dongguan branch)

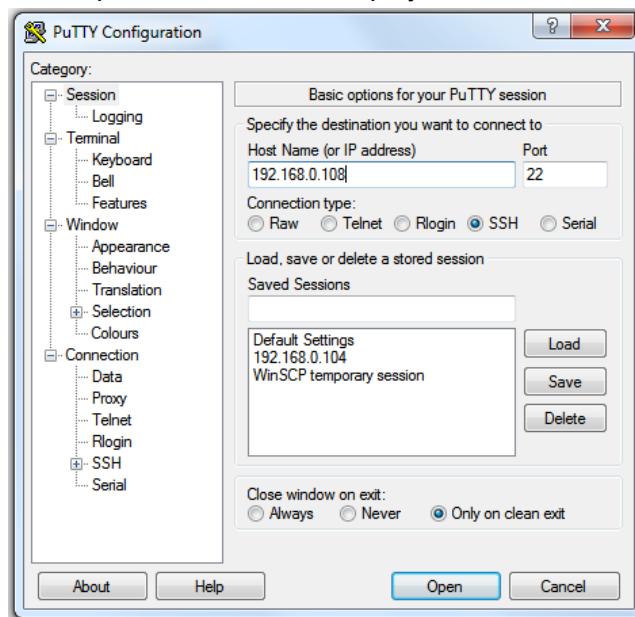
Nmap scan report for 192.168.0.100
Host is up <0.020s latency>
All 1000 scanned ports on 192.168.0.100 are closed
MAC Address: 08:AA:55:7B:9C:A3 (Motorola Mobility, a Lenovo Company)

Nmap scan report for 192.168.0.108
Host is up <0.0066s latency>.
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5900/tcp  open  vnc
MAC Address: B8:27:EB:4C:B3:17 (Raspberry Pi Foundation)

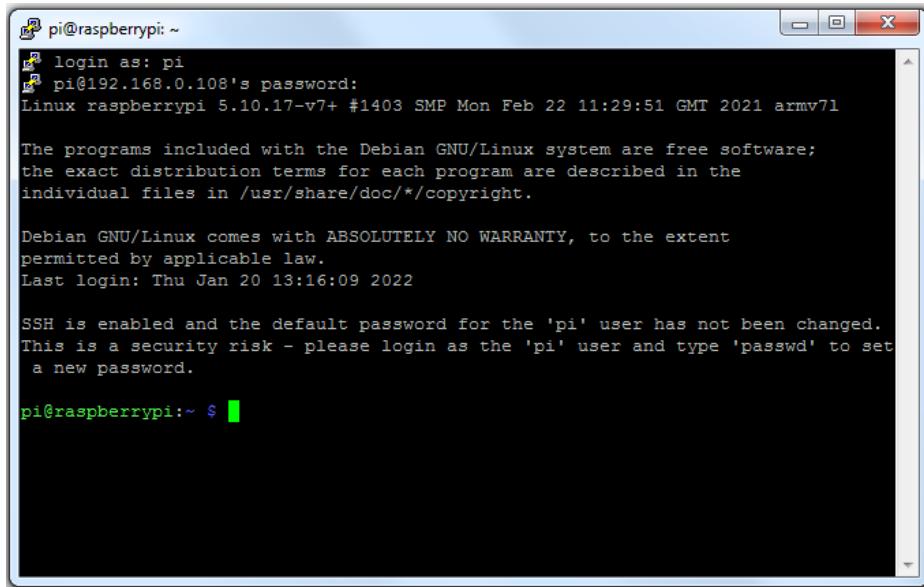
Nmap scan report for 192.168.0.112
Host is up <0.039s latency>.
Not shown: 995 closed ports
PORT      STATE SERVICE
3000/tcp  open  ppp
3001/tcp  open  nessus
7778/tcp  open  interwise
9988/tcp  open  glrp
9998/tcp  open  distinct32
MAC Address: 30:A9:DE:25:50:D2 (LG Innotek)

Nmap scan report for 192.168.0.118
Host is up <0.020s latency>
All 1000 scanned ports on 192.168.0.118 are closed
MAC Address: 8A:A8:9D:08:32:10 (Unknown)
```

Si lo que desea es acceder a la terminal de comandos se recomienda el programa Putty, el cual le permitirá acceder de forma remota a la terminal del dispositivo con su IP y su puerto. Solicitará un usuario y contraseña que por defecto en estas placas el usuario es ‘pi’ y la contraseña ‘raspberry’.

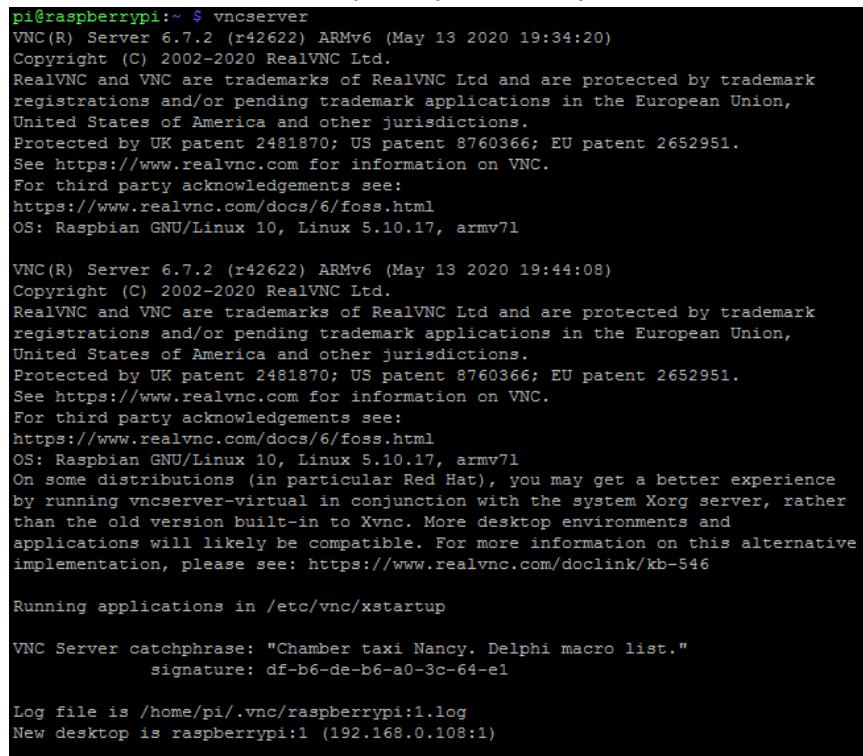


Deberá aparecer la línea de comandos del dispositivo:



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the standard Raspbian login screen with the following text:
pi@raspberrypi: pi
pi@192.168.0.108's password:
Linux raspberrypi 5.10.17-v7+ #1403 SMP Mon Feb 22 11:29:51 GMT 2021 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 20 13:16:09 2022
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.
pi@raspberrypi: ~

Si desea acceder a la interfaz gráfica se recomienda el uso del programa **VNC Viewer**. Ejecute **vncserver** desde la línea de comandos, lo que le permitirá ejecutar un servidor vnc.



A screenshot of a terminal window titled "pi@raspberrypi: ~ \$ vncserver". The window shows the output of the vncserver command:
pi@raspberrypi: ~ \$ vncserver
VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:34:20)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See <https://www.realvnc.com> for information on VNC.
For third party acknowledgements see:
<https://www.realvnc.com/docs/6/foss.html>
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l

VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:44:08)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See <https://www.realvnc.com> for information on VNC.
For third party acknowledgements see:
<https://www.realvnc.com/docs/6/foss.html>
OS: Raspbian GNU/Linux 10, Linux 5.10.17, armv7l
On some distributions (in particular Red Hat), you may get a better experience
by running vncserver-virtual in conjunction with the system Xorg server, rather
than the old version built-in to Xvnc. More desktop environments and
applications will likely be compatible. For more information on this alternative
implementation, please see: <https://www.realvnc.com/doclink/kb-546>

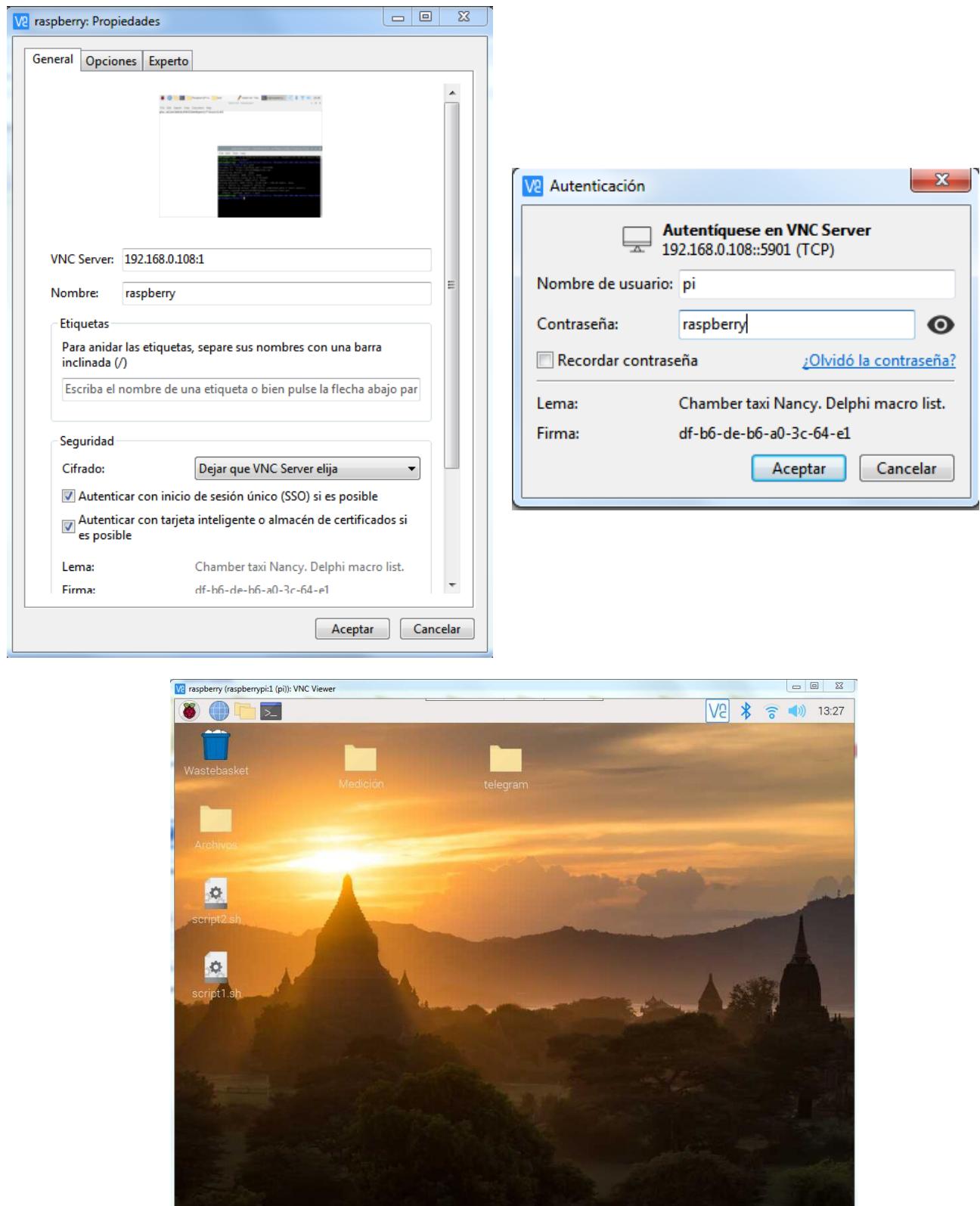
Running applications in /etc/vnc/xstartup

VNC Server catchphrase: "Chamber taxi Nancy. Delphi macro list."
signature: df-b6-de-b6-a0-3c-64-e1

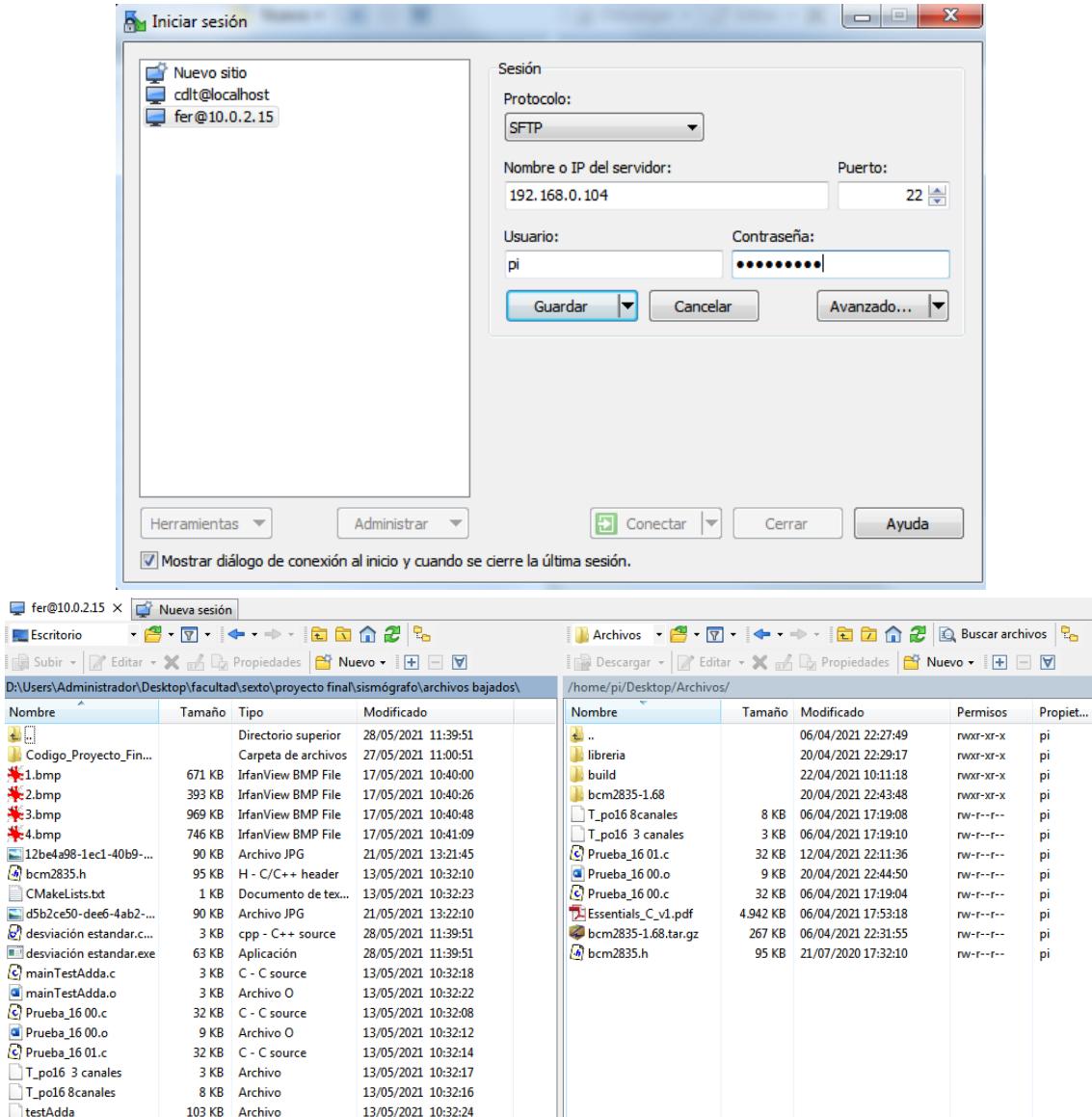
Log file is /home/pi/.vnc/raspberrypi:1.log
New desktop is raspberrypi:1 (192.168.0.108:1)

Generará un nuevo escritorio con la dirección IP asignada a la placa seguido de dos puntos y un número.

En el programa VNC Viewer en su computadora coloque el escritorio generado por el servidor; también pedirá un identificador y luego el usuario y contraseña que son las ya mencionadas anteriormente. De esta forma podrá acceder al escritorio de la placa Raspberry Pi como si la hubiera conectado de forma directa a un monitor, teclado y mouse.



Por último, si desea descargar los archivos generados con las muestras, se recomienda utilizar el programa WinSCP, que permite realizar la transferencia de archivos de una forma muy cómoda utilizando la dirección IP y el puerto asignado a la placa.



Observe que se presentan dos pantallas, una con los directorios de su PC y la otra con los de la Raspberry Pi. Simplemente copiando y pegando de una pantalla a otra podrá descargar los archivos generados.

➤ Fuera de la red de área local.

Puede presentarse la necesidad de descargar los archivos o modificar los parámetros sin que el usuario se encuentre conectado a la misma red que el dispositivo. Para esto se utiliza el servicio de mensajería de Telegram, y particularmente la interfaz Telegram Bot API.

Los pasos y experiencia se detallan en la sección **Chat de Telegram**.

11.3 Parámetros de configuración

El usuario puede modificar parámetros que le permiten controlar ciertos aspectos y límites a la hora de realizar las mediciones y generar los archivos. Estos parámetros son los siguientes:

1) Canal de disparo:

Con este parámetro el usuario puede seleccionar cuál de los 8 canales de medición provocará el guardado de las muestras tomadas en un nuevo archivo, si se supera el valor umbral especificado en nivel de disparo. Los canales se enumeran del 0 al 7. En caso de ingresar un número mayor a 7, el parámetro tomará el número 7 por defecto. En caso de ingresar un número menor a 0, el parámetro tomará el número 0 por defecto.

2) Frecuencia de muestreo:

Al modificar la frecuencia de muestreo se modifica la velocidad con la que el dispositivo toma las sucesivas muestras. A mayor frecuencia de muestreo, mayor velocidad en la toma de mediciones.

Existe un amplio abanico de posibles frecuencias de muestreo seleccionables desde un mínimo de 2.5 SPS (Samples Per Second), hasta un máximo de 30000 SPS. Es importante aclarar que, aunque estas sean las frecuencias de muestreo especificadas en la hoja de datos del conversor ADS1256, la frecuencia real de muestreo es mucho menor debido a la comutación de canales, los tiempos de espera necesarios para que el conversor pueda tomar muestras de forma correcta, etc.

Estas frecuencias son:

- 30000 SPS.
- 15000 SPS.
- 7500 SPS.
- 3750 SPS.
- 2000 SPS.
- 1000 SPS.
- 500 SPS.
- 100 SPS.
- 60 SPS.
- 50 SPS.
- 30 SPS.
- 25 SPS.
- 15 SPS.
- 10 SPS.
- 5 SPS.
- 2.5 SPS.

Si decide utilizar los 8 canales de medición, la frecuencia real a 30000 SPS será de aproximadamente 390 muestras/segundo.

Debido a que el conversor solo puede trabajar a las frecuencias ya mencionadas, evite ingresar otros números que no se correspondan al listado de frecuencias mostrado. Caso contrario el conversor no interpretará correctamente a qué frecuencia debe muestrear.

3) Nivel de disparo:

Este es el valor umbral que es comparado constantemente con la muestra del canal de disparo. En caso de que alguna muestra iguale o supere este valor umbral, se detiene la medición de los canales y se guardan las muestras en el archivo.

Se ingresa en el siguiente formato:

“U.dc” en volts.

Por ejemplo: 1.38 V

Los límites que puede adoptar este valor dependerán del rango seleccionado en base a los jumpers de entrada en el circuito de protección y de si el buffer analógico de entrada del conversor se encuentra habilitado o deshabilitado.

Con el buffer deshabilitado el rango de tensión de entrada se encuentra entre -0.1 V y 5.1 V.

Con el buffer habilitado el rango de tensión de entrada se encuentra entre 0 V y 3 V.

4) Cantidad de muestras post-disparo:

Una vez superado el nivel de disparo, el dispositivo seguirá tomando una determinada cantidad de muestras. Esta cantidad es la especificada en este parámetro. De esta forma si el usuario desea 100 muestras luego del disparo, el dispositivo utiliza este parámetro para tomar 100 muestras más de todos los canales antes de realizar el guardado en el archivo correspondiente.

Debido a que las muestras se almacenan en arreglos enteros y la memoria del dispositivo es limitada, existe un límite en el valor que este parámetro puede adoptar. Este es de 39000 muestras si la cantidad de muestras pre-disparo es 0. Por precaución, en caso de que la suma de la cantidad de muestras post disparo y pre-disparo superen las 39000 muestras, ambos parámetros adoptarán el valor de 18000 muestras, dando un total de 36000 muestras. Este valor se ha tomado teniendo en cuenta un margen de seguridad.

5) Cantidad de muestras pre-disparo:

De la misma forma en que pueden almacenarse muestras posteriores al disparo, pueden almacenarse muestras anteriores a este. Esto puede hacerse debido a que el dispositivo mide constantemente y almacena las muestras en un buffer que es el arreglo ya mencionado. De esta manera, si el usuario desea 100 muestras anteriores al disparo, estas ya se encuentran almacenadas. En caso de que el disparo se produzca al poco tiempo de haber iniciado el dispositivo y si no se poseen muestras anteriores suficientes para alcanzar a la cantidad especificada, las muestras faltantes toman el valor 0.

También el límite que puede adoptar este parámetro es de 39000 muestras, aunque como ya se ha mencionado, en caso de que la suma de la cantidad de muestras post disparo y pre-disparo superen las 39000 muestras, ambos parámetros adoptarán el valor de 18000 muestras

6) Cantidad de archivos almacenables:

Este parámetro se utiliza para especificar la cantidad máxima de archivos que se generarán antes de comenzar a sobrescribir a los archivos más nuevos. De esta forma, si el usuario desea que se almacenen solo 100 archivos, una vez llegado al archivo número 102 (debido a que el primer archivo es el número 0), este nuevo archivo tomará el valor ‘000’ y sobrescribirá al archivo más antiguo, reiniciándose el conteo de estos.

Consideración: no existe limitación para la cantidad de archivos que pueden almacenarse en un directorio, por lo tanto, el límite lo fijará la cantidad de memoria disponible.

Estos parámetros están contenidos en un archivo de configuración de texto llamado *Parametros.txt* el cual se encuentra en ***home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build***. El usuario puede abrir este archivo y modificar el parámetro que desee, o acceder a dichos parámetros de forma remota. Los mismos poseen el siguiente formato:

1-Canal-4

2-Frecuencia-30000

3-Nivel-1.57
4-Muestras posttrigger-12000
5-Muestras pretrigger-673
6-Cantidad de archivos-1000

#Consideraciones#

*La suma de la cantidad de muestras pre y post disparo no debe superar las 39000 muestras.

*La frecuencia de muestreo real solo se aproxima a la seleccionada en el caso de leer un solo canal. Para 8 canales la frecuencia máxima es de aproximadamente 390 muestras/segundo.

*El nivel máximo admisible en las entradas analógicas no debe superar los 5 V con el buffer de entrada deshabilitado, por lo que el nivel de disparo debe ser menor.

*Los canales de lectura van del número 0 al número 7.

El programa lee y obtiene estos parámetros al inicio de este, pero los mismos pueden ser aplicados nuevamente de forma remota a través de una de las opciones ofrecidas por la interfaz de Telegram.

Ya que el programa lee automáticamente estos parámetros al inicio, en caso de que el dispositivo se quede sin alimentación, al reiniciar no se requiere que exista un usuario presente en el lugar que deba ingresar los parámetros en forma manual, directamente los obtiene nuevamente de este archivo.

En caso de que este archivo de configuración no pueda abrirse o se dañe, el programa intentará obtener los parámetros de un segundo archivo de configuración de respaldo llamado *Parametros_respaldo.txt*, el cual contiene los siguientes parámetros por defecto:

1-Canal-0
2-Frecuencia-30000
3-Nivel-1.50
4-Muestras posttrigger-10000
5-Muestras pretrigger-10000
6-Cantidad de archivos-1000

Precaución: al modificar los parámetros en el archivo de configuración, evite modificar cualquier otro carácter que no sea el número del parámetro. Caso contrario el programa podría obtener incorrectamente el parámetro modificado debido a que utiliza la posición relativa al inicio de cada línea del archivo para obtener el parámetro en cuestión.

Consideración: Tanto si se interrumpe el programa mientras se tiene abierto el archivo *parámetros.txt* como *lastfile.txt*, los mismos no se ven afectados. Lo mismo ocurre si se corta la alimentación.

Puede modificar el archivo *Parametros.txt* mientras el programa se encuentra en ejecución.

11.4 Manejo de archivos

La generación de archivos se realiza en forma automática cada vez que alguna muestra tomada por el canal seleccionado como canal de disparo supere el valor umbral también seleccionado por el usuario. Los

archivos se generarán en el directorio ***home/pi/Desktop/Archivos/libreria /RaspberryPi-ADC-DAC-master/build***.

Cada archivo poseerá un número junto con la fecha, hora y minuto en que se generó. El formato es el siguiente:

“00XXXX-DD|MM|AAAA-HH:MM:SS”

XXXX representa el número de archivo, cuya longitud variará a medida que se generen nuevos archivos.

Los archivos generados tienen formato de archivo de texto y poseen un encabezado con:

- Los parámetros de configuración.
- El número de archivo.
- La fecha de generación.
- El factor de escala.

Cada columna representa un canal de medición y novena y décima columna son las estampas de tiempo de cada muestra en segundos y microsegundos.

Un archivo típico puede verse a continuación:

Wed Dec 29 19:43:54 2021	Parámetros:	Canal de disparo: 1	Frecuencia de muestreo: 30000	Nivel de disparo=	1.500000	Cantidad de			
muestras post trigger =	500	Cantidad de muestras pre trigger =	300	Número de archivo =	15	Volts/cuenta =			
5000000/8388608	Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)			
0	0	0	0	0	0	0			
0	0	0	0	0	0	0			
0	0	0	0	0	0	0			
0	0	0	0	0	0	0			
0	0	0	0	0	0	0			
832614	847677	862886	877447	892226	906783	922021	938248	1634741529	124592
953393	969037	984859	1000830	1016420	1032779	1049411	1066684	1634741529	127208
1085099	1102879	1120202	1137461	1155041	1172410	1189801	1208170	1634741529	129830
1225322	1243005	1261040	1278634	1296227	1313563	1331614	1350188	1634741529	132460
1367667	1385324	1403622	1422291	1441035	1459607	1478573	1499416	1634741529	135051
1519315	1539355	1559987	1579621	1598739	1618177	1638444	1659028	1634741529	137683
1678993	1699629	1720506	1741522	1762194	1782079	1802074	1823208	1634741529	140284
1843338	1863530	1884241	1904617	1924334	1944174	1964490	1985684	1634741529	142957
2005789	2025728	2046384	2066537	2086711	2107103	2127402	2148816	1634741529	145557
2169009	2189696	2211146	2232452	2253774	2274842	2296045	2318346	1634741529	148125
2339663	2361208	2383551	2405094	2426776	2448742	2470466	2493474	1634741529	150693
2515069	2536998	2559469	2581169	2602698	2623782	2644444	2666311	1634741529	153298
2689381	2712032	2733638	2754879	2775975	2798054	2819921	2843088	1634741529	156015
2867229	2889157	2910647	2931732	2952296	2972616	2993181	3014776	1634741529	158672
3036566	3057008	3077750	3098698	3119124	3139376	3160254	3181228	1634741529	161321
3205515	3225717	3245669	3265581	3285510	3305295	3325251	3345567	1634741529	164015
3364978	3385932	3406173	3426700	3446782	3467074	3487035	3508123	1634741529	166648
3529353	3549660	3568846	3587298	3605445	3624236	3642776	3661893	1634741529	169388
3680034	3697814	3715796	3733408	3751190	3768668	3786260	3804678	1634741529	171991
3822210	3840224	3857678	3874568	3891197	3908464	3925627	3942987	1634741529	174593

El cual, al ser exportado a un archivo de Excel queda:

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10
Wed Dec 29 19:43:54 2021									
Parámetros:									
Canal de disparo:	1								
Frecuencia de muestreo:	30000								
Nivel de disparo=	1.500000								
Cantidad de muestras post trigger =	500								
Cantidad de muestras pre trigger =	300								
Número de archivo =	15	Volts/cuenta = 5000000/8388608							
Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) + uS	
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
832614	847677	862886	877447	892226	906783	922021	938248	1634741529	124592
953393	969037	984859	1000830	1016420	1032779	1049411	1066684	1634741529	127208
1085099	1102879	1120202	1137461	1155041	1172410	1189801	1208170	1634741529	129830
1225322	1243005	1261040	1278634	1296227	1313563	1331614	1350188	1634741529	132460
1367667	1385324	1403622	1422291	1441035	1459607	1478573	1499416	1634741529	135051
1519315	1539355	1559987	1579621	1598739	1618177	1638444	1659028	1634741529	137683
1678993	1699629	1720506	1741522	1762194	1782079	1802074	1823208	1634741529	140284
1843338	1863530	1884241	1904617	1924334	1944174	1964490	1985684	1634741529	142957
2005789	2025728	2046384	2066537	2086711	2107103	2127402	2148816	1634741529	145557
2169009	2189696	2211146	2232452	2253774	2274842	2296045	2318346	1634741529	148125
2339663	2361208	2383551	2405094	2426776	2448742	2470466	2493474	1634741529	150693
2515069	2536998	2559469	2581169	2602698	2623782	2644444	2666311	1634741529	153298
2689381	2712032	2733638	2754879	2775975	2798054	2819921	2843088	1634741529	156015
2867229	2889157	2910647	2931732	2952296	2972616	2993181	3014776	1634741529	158672
3036566	3057008	3077750	3098698	3119124	3139376	3160254	3181228	1634741529	161321
3205515	3225717	3245669	3265581	3285510	3305295	3325251	3345567	1634741529	164015

Cada muestra se encuentra expresada en cuentas arrojadas por el conversor. Si desea obtener su valor equivalente en volts, multiplíquela por el factor de escala Volts/cuenta.

Recuerde que, si la cantidad de muestras almacenadas al momento de producirse el disparo no es suficiente para alcanzar la cantidad de muestras pre-disparo, las muestras faltantes al igual que sus estampas de tiempo se llenan con ceros.

La cantidad de archivos almacenados estará limitada por la cantidad de archivos almacenables, parámetro modificable por el usuario. Por lo tanto, el primer archivo numérico será el 000 y el último variará en función de este parámetro. Una vez que se supere esta cantidad, los nuevos archivos sobreescibirán a los más antiguos. Por ejemplo, si la cantidad de archivos almacenables seleccionada por el usuario es de 1000, se almacenarán 1001 archivos, desde el 000 hasta el 1000. El archivo 1002 sobreescibirá al número 000, el 1003 al 001 y así sucesivamente.

Para descargar estos archivos a una computadora puede hacerlo a través del programa WinSCP, que permite realizar la transferencia de archivos utilizando la dirección IP y el puerto asignado a la placa. Puede ver el procedimiento completo en la sección **2. Encendido y acceso al dispositivo**.

También puede descargar los archivos remotamente a través de Telegram. Las opciones de visualización y descarga se detallan en la sección **5. Chat de Telegram**.

Puede determinar cuál es el último archivo generado bien a través de la fecha contenida en el nombre del archivo o bien a través del archivo *lastfile.txt*. Este archivo contiene el número del siguiente archivo a generar, por lo que si, por ejemplo, contiene el número 10, el último archivo generado es el número 9. Procure no modificar este archivo a menos que desee reiniciar o cambiar el conteo de archivos.

Consideraciones:

- Si se modifica el parámetro de cantidad de archivos almacenables de una cantidad mayor a una menor, si existen los archivos de mayor numeración a la nueva cantidad especificada, estos no se eliminarán a menos que sean removidos manualmente accediendo a la placa. Para entender mejor

la situación se dará un ejemplo. Suponga que originalmente la cantidad de archivos almacenables era 100 y se han generado los 100 archivos. Si se modifica esta cantidad a 50, al sobreescibirse los primeros archivos solo lo harán hasta el número 50, ya que esta es la nueva cantidad máxima a almacenar. Por lo tanto, los archivos cuyo número esté comprendido entre el 51 y el 100 no se sobreescibirán y quedarán almacenados hasta que el usuario acceda a la placa para eliminarlos o descargarlos.

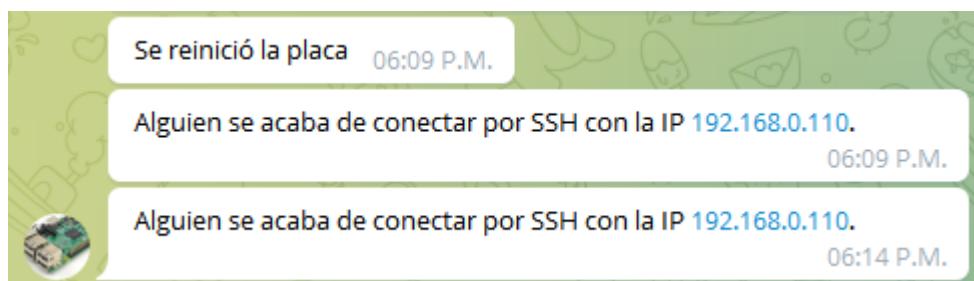
- *La sobreescritura de archivos se realiza primero eliminando el archivo antiguo y luego generando el nuevo archivo con el mismo número. En caso de que se interrumpa la alimentación luego de eliminar el archivo antiguo, pero antes de generar el nuevo, simplemente existirá un número de archivo faltante hasta que se genere un nuevo archivo que ocupe dicho número.*

11.5 Chat de Telegram

La forma de acceder remotamente al dispositivo sin necesidad de que se encuentre dentro de la misma red que este, es a través del servicio de mensajería de Telegram. Particularmente se hace uso de la interfaz Telegram Bot API, la cual es una interfaz de programación de Telegram orientada al uso de bots.

A través del uso de bots podrá acceder al listado de los archivos generados, descarga de estos, modificación de los parámetros de configuración, etc.

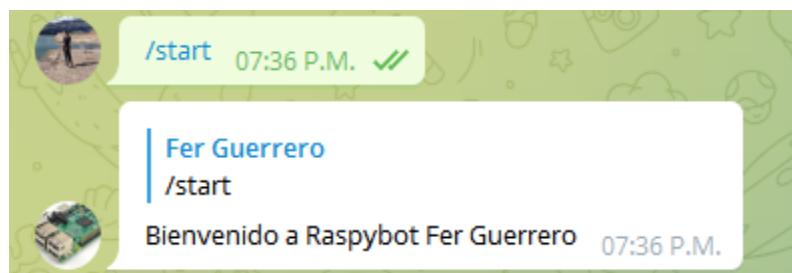
Si ya ha creado su bot y seguido los pasos de instalación, cada vez que el sistema operativo se inicie y acceda a la terminal de comandos llegará a su chat el siguiente mensaje:



Aparecerán los mensajes de que se ha realizado una conexión SSH cada vez que ingrese a través de la terminal, o desde el programa PuTTY, indicando la dirección IP del usuario conectado.

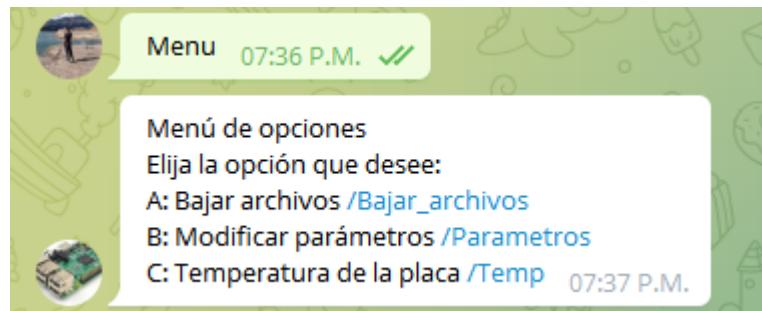
La interacción con el bot se realiza a través de menús y comandos.

Si envía el comando **/start** recibirá un mensaje de bienvenida:



- **Menú principal:**

Para acceder al menú principal debe enviar la palabra “Menu” (sin acento):



Evite el uso de acentos debido a que suelen causar errores en el envío de mensajes.

Note que los comandos se muestran en color celeste. Puede cliquear sobre estos comandos como si de un botón se tratara e inmediatamente se enviará el comando correspondiente.

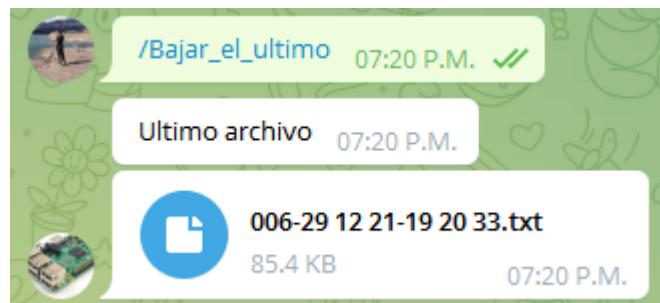
- **Opción A: Bajar archivos**

Al elegir la opción A para la descarga de archivos, se presenta el siguiente submenú:



➤ **Bajar el último archivo generado:**

Para descargar el archivo más nuevo generado, elija la opción número 1 a través del comando **/Bajar_el_ultimo**.

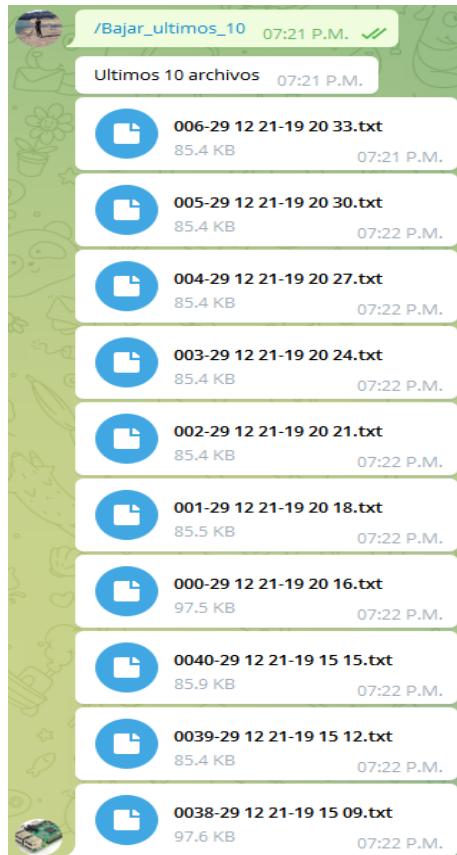


Al cliquear sobre el archivo, automáticamente se descargará:

Wed Dec 29 19:20:33 2021									
Parámetros:									
Canal de disparo:	4								
Frecuencia de muestreo:	15000								
Nivel de disparo=	2.00								
Cantidad de muestras post trigger =	1000								
Cantidad de muestras pre trigger =	200								
Número de archivo =	6								
Volts/cuenta =	5000000/8388608								
Canal 0 (uV)	Canal 1 (uV)	Canal 2 (uV)	Canal 3 (uV)	Canal 4 (uV)	Canal 5 (uV)	Canal 6 (uV)	Canal 7 (uV)	Tiempo: canal 0 (S) +	uS
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
4127430	4125379	4112659	4112270	4111199	4109368	4107854	4105699	1640816430	408650
4101519	4094235	4085622	4077752	4069688	4061174	4056468	4050241	1640816430	411958
4046861	4045583	4044252	4043050	4042059	4041589	4041432	4042123	1640816430	414796
4042970	4044032	4045153	4046119	4047512	4049512	4050889	4053656	1640816430	417629
4055898	4058294	4060508	4062704	4067919	4077205	4083847	4095390	1640816430	420475
4104369	4112057	4118274	4122735	4126488	4129620	4130815	4133204	1640816430	423312
4134517	4135326	4135240	4134233	4133051	4131521	4130107	4127887	1640816430	426151
4125617	4123324	4120656	4118285	4115533	4112679	4110513	4107619	1640816430	428986
4104532	4096735	4087989	4079854	4071303	4062027	4057186	4050394	1640816430	431830
4046229	4044700	4043271	4042113	4040970	4040239	4039978	4040595	1640816430	434668
4041464	4042512	4043739	4044729	4046093	4048180	4049508	4052482	1640816430	437507
4054859	4057446	4059844	4061989	4066315	4076004	4082862	4095024	1640816430	440363
4104649	4113331	4119926	4124907	4129214	4132879	4134238	4136862	1640816430	443182
4138399	4139429	4139588	4138648	4137390	4135694	4134203	4131773	1640816430	446019

➤ Bajar los últimos 10 archivos generados:

Para descargar los 10 archivos más nuevos generados, elija la opción número 2 a través del comando **/Bajar_ultimos_10**.



Se descargarán en orden del archivo más nuevo al más antiguo. Puede observar que, aunque llegue al archivo número 0, de existir, seguirá por el mayor archivo numérico.

Si se ha accedido desde su celular, puede aparecer el siguiente mensaje:

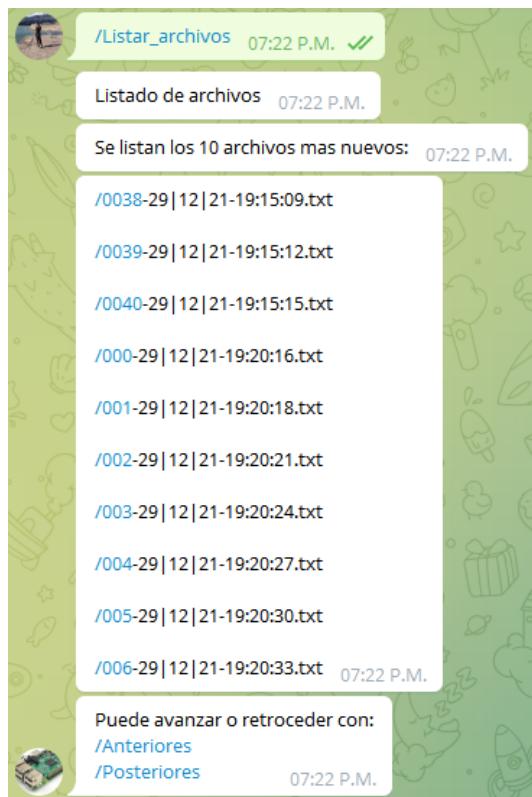


En dicho caso elija la opción subir y podrá visualizar el archivo.

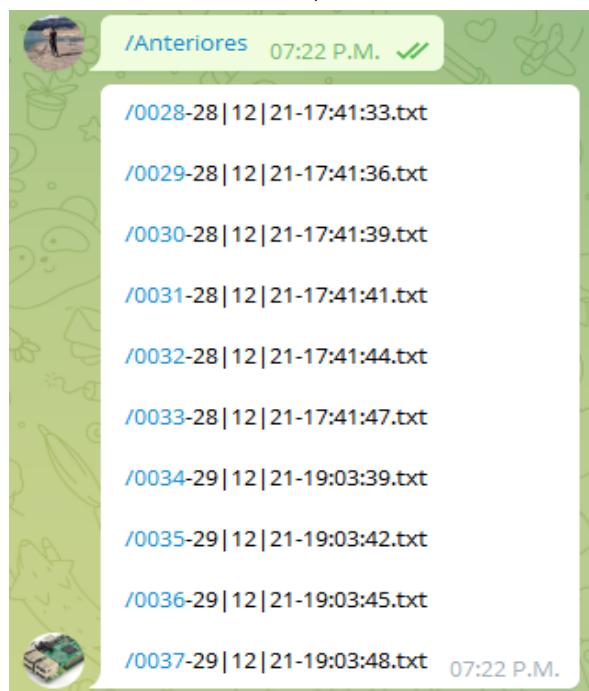
➤ **Listar los archivos:**

Si desea un listado de los archivos almacenados, elija la opción número 3 a través del comando **/Listar_archivos**. Primeramente, se presentará un listado de los últimos 10 archivos y las opciones de listar los archivos anteriores y posteriores.

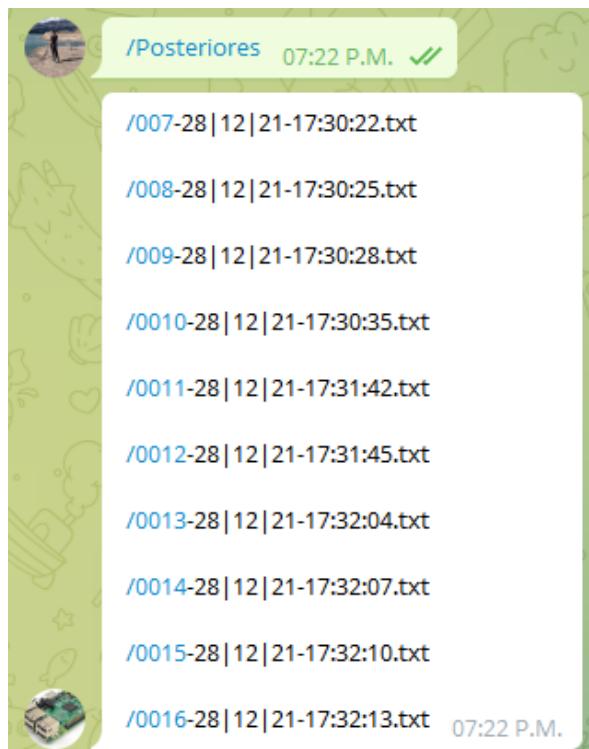
Los archivos serán listados en orden del más antiguo al más nuevo.



Si desea listar los 10 archivos numéricos anteriores, utilice el comando **/Anteriores**:

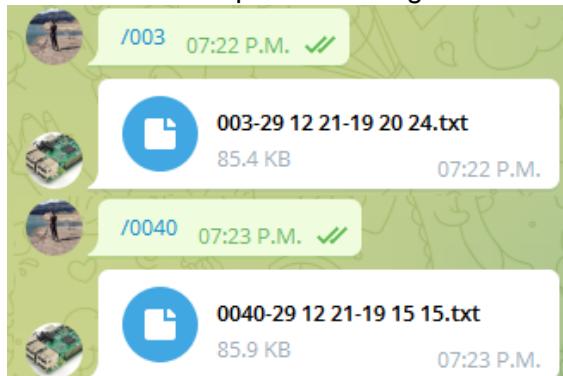


Si desea listar los 10 archivos numéricos posteriores, utilice el comando **/Posteriores**:



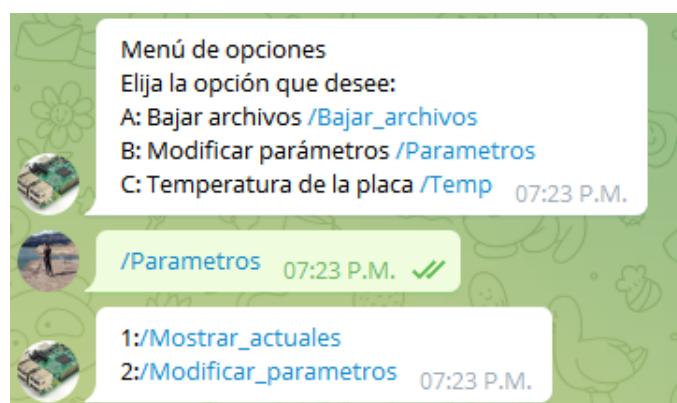
Puede avanzar y retroceder las veces que desee para visualizar el listado completo de los archivos almacenados.

Cliqueando sobre los números de los archivos podrá descargar individualmente el archivo que desee:



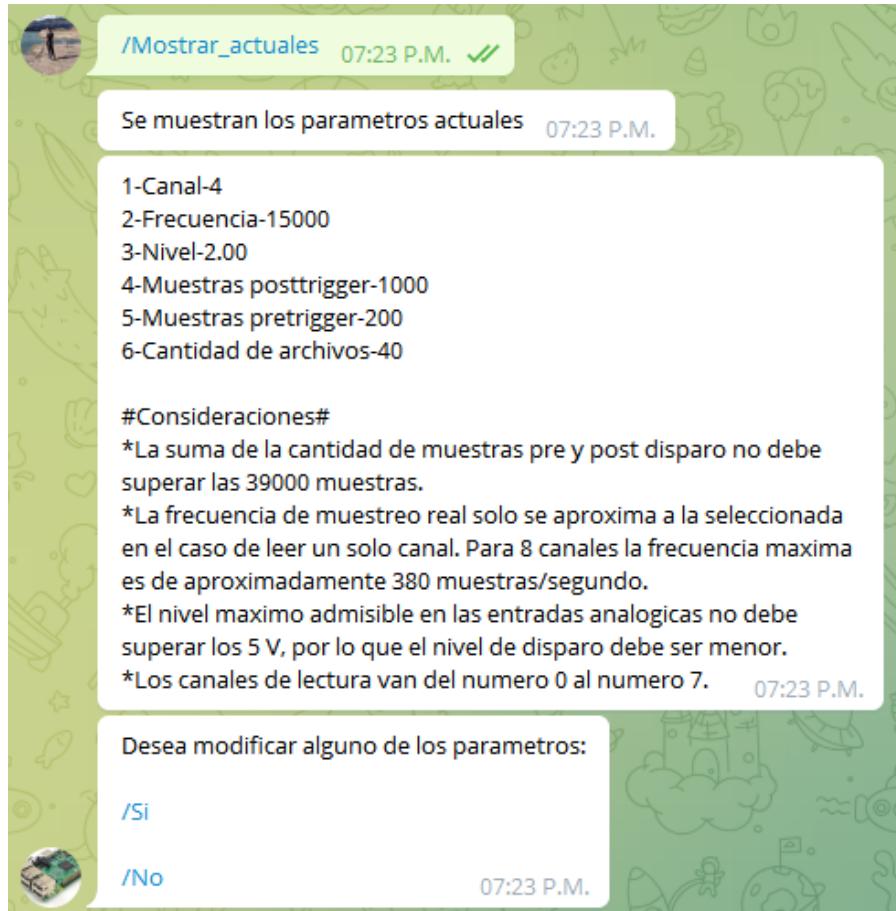
○ **Opción B: Modificar parámetros**

Si desea modificar los parámetros de configuración elija la opción B a través del comando **/Parametros**. Se presentará el siguiente submenú:



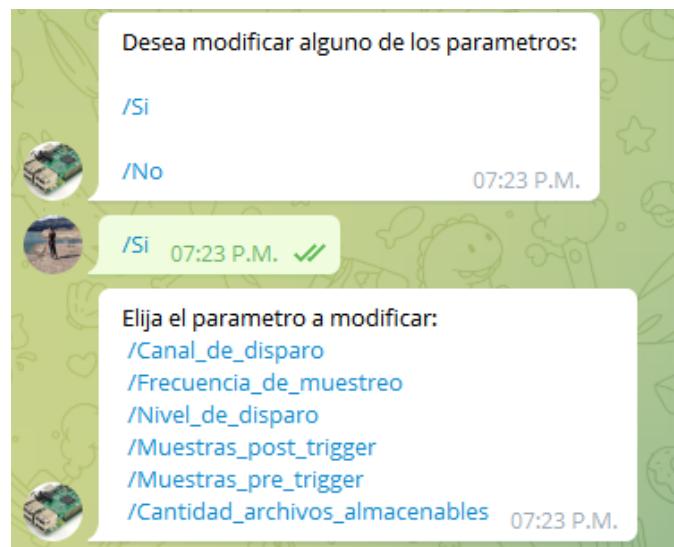
➤ **Mostrar los parámetros actuales:**

Si desea visualizar los parámetros actuales de configuración, elija la opción 1 a través del comando **/Mostrar_actuales**.



Se muestran además las consideraciones que deben tenerse en cuenta en la modificación de estos parámetros.

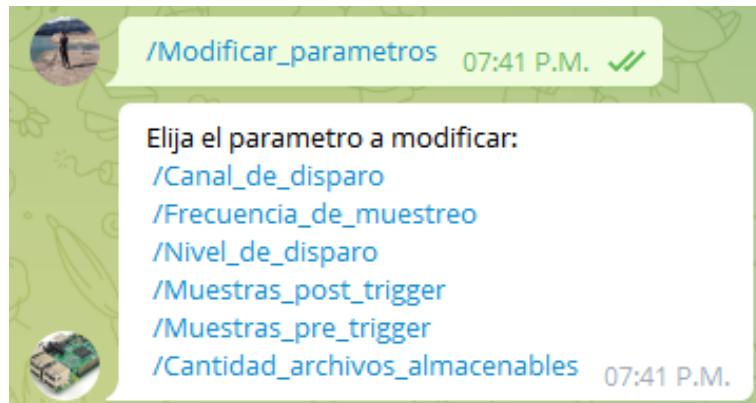
Adicionalmente se presenta la opción de modificar estos parámetros. Si envía el comando `/Si`, se le mostrará el menú con los parámetros a modificar:



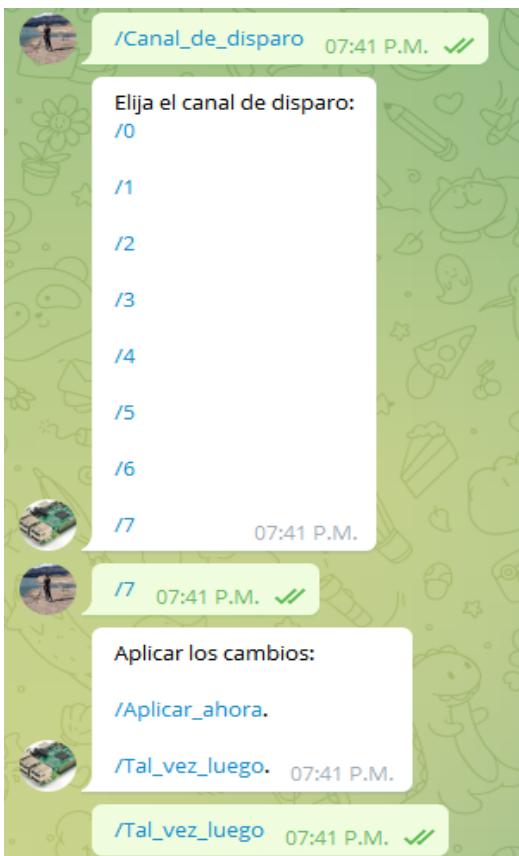
Este menú es el mismo que se muestra en caso de elegir la opción 2 de modificar los parámetros, solo si el comando **/Si** es enviado luego de haber mostrado los parámetros actuales. Si envía el comando **/No**, no se mostrará ninguna nueva opción, pero cualquier comando posterior **/Si** no tendrá efecto en la modificación de estos parámetros.

➤ **Modificar los parámetros:**

Si lo que desea es modificar los parámetros de configuración elija la opción 2 a través del comando **/Modificar_parametros**:



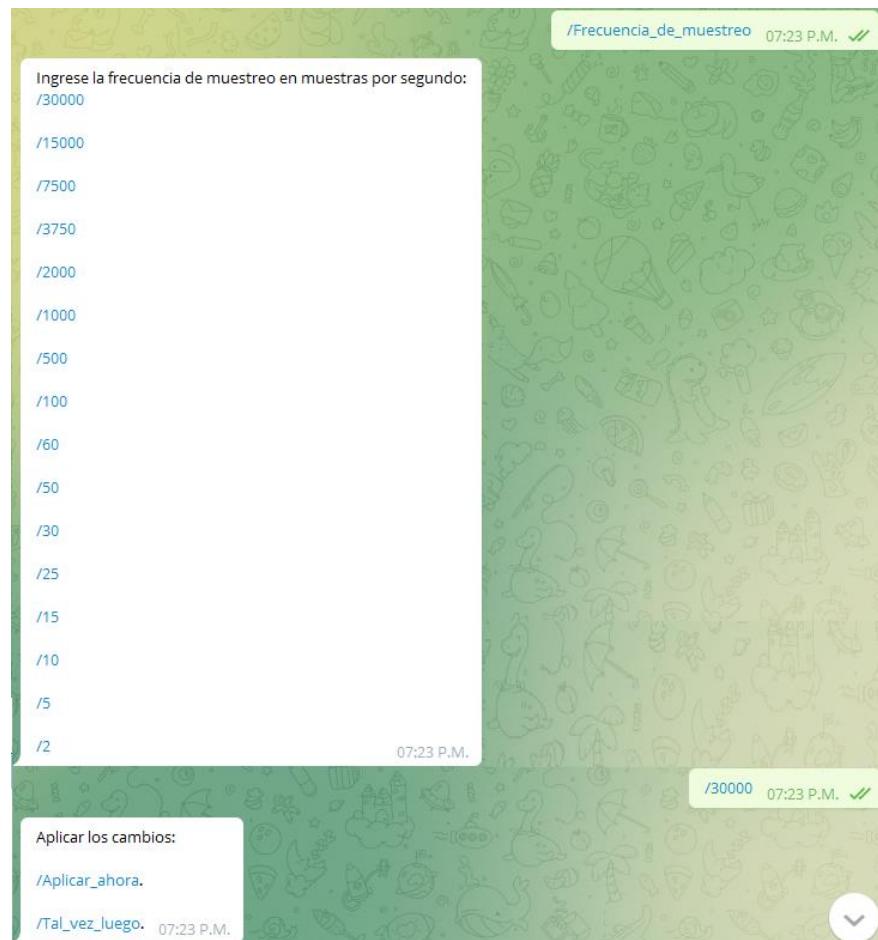
1. **Canal de disparo:** a través del comando **/Canal_de_disparo** puede seleccionar el canal que se monitoreará evaluando si alguna de sus mediciones supera le valor umbral de disparo y desencadena la generación de archivos.



Se presentan los 8 canales del 0 al 7. Una vez seleccionado el canal, le aparecerá la opción de aplicar los cambios realizados o hacerlo luego. En caso de colocar el comando **/Aplicar_ahora** se envía automáticamente una señal al programa de medición para leer nuevamente los parámetros y ser aplicados donde correspondan. Si coloca el comando **/Tal_vez_luego**, simplemente se habrá modificado el archivo de configuración, pero el programa de medición seguirá utilizando los parámetros previos.

Estas opciones se muestran en cada parámetro.

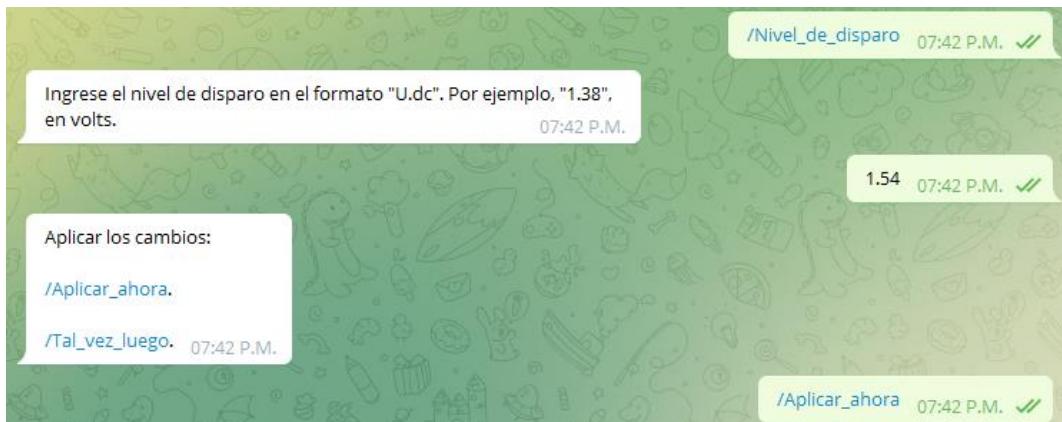
2. Frecuencia de muestreo: si desea modificar la frecuencia de muestreo, envíe el comando **/Frecuencia_de_muestreo:**



Recordar que, aunque se muestren las frecuencias desde 2.5 SPS a 30000 SPS, la frecuencia de muestreo real es menor como se mencionó anteriormente.

Aunque se envíe el comando **/2** la frecuencia de muestreo real será de 2.5 SPS.

3. Nivel de disparo: para modificar el nivel umbral de disparo envíe el comando **/Nivel_de_disparo**:



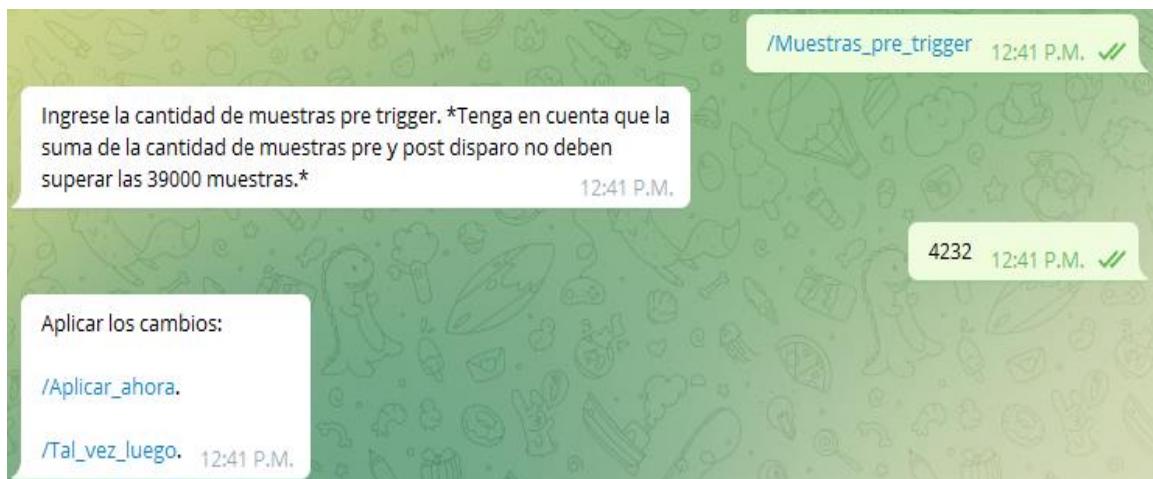
El nivel debe ser ingresado en volts en el formato de una unidad y dos decimales.

4. Cantidad de muestras post disparo: para modificar la cantidad de muestras luego del disparo envíe el comando **/Muestras_post_trigger**:

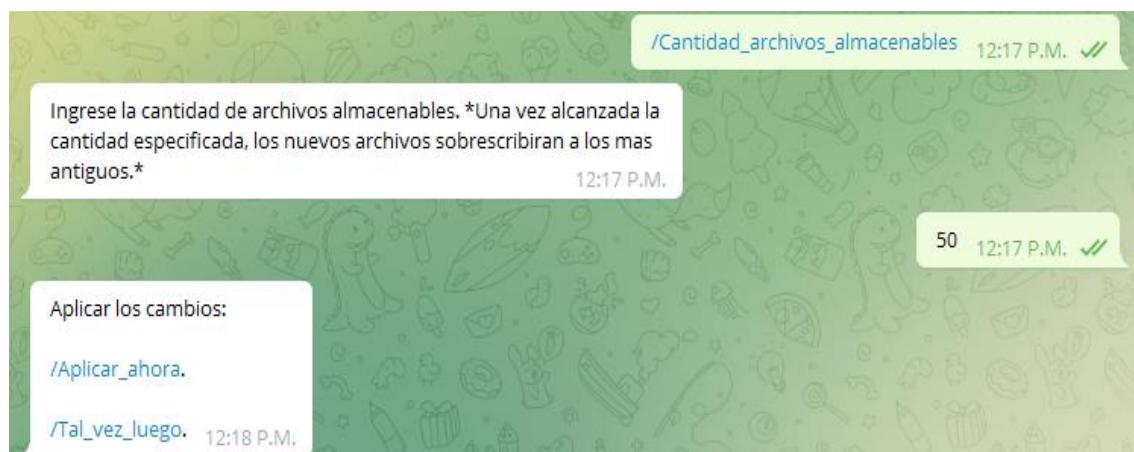


Si la suma entre la cantidad de muestras post y pre disparo supera las 39000 muestras, se limitarán ambas cantidades a 18000 muestras.

5. **Cantidad de muestras pre disparo:** para modificar la cantidad de muestras antes del disparo envíe el comando **/Muestras_pre_trigger**:

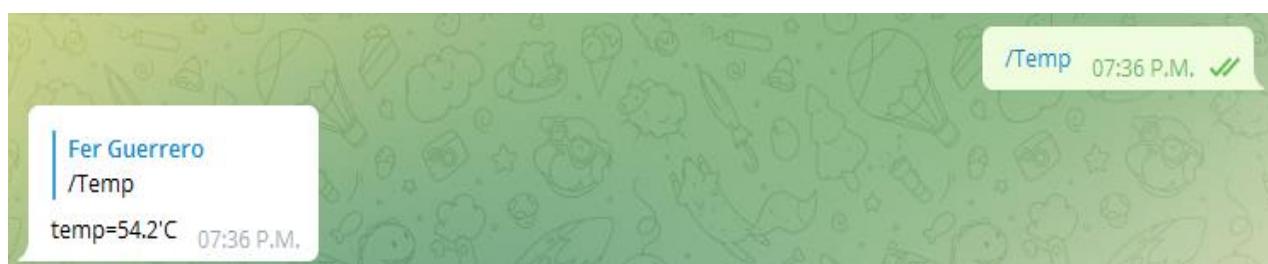


6. **Cantidad de archivos almacenables:** para modificar la cantidad de archivos que se generarán antes de comenzar a sobrescribirlos, envíe el comando **/Cantidad_archivos_almacenable**s:



■ Opción C: Visualizar la temperatura de la placa

La tercera opción presentada en el menú principal es la de testear la temperatura interna de la placa Raspberry Pi. Para ello utilice el comando **/Temp**:



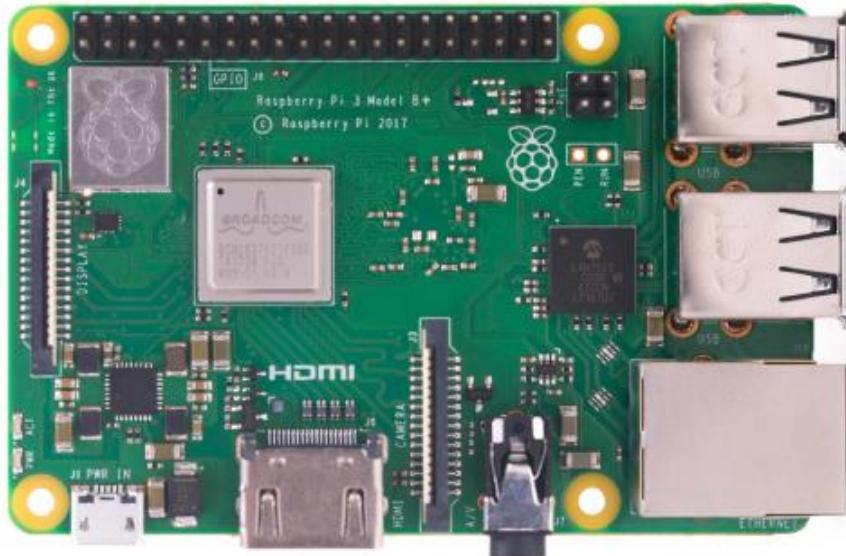
Listado general de comandos:

- ❖ **/start:** recibirá un mensaje de bienvenida.
- ❖ **Menu:** no es un comando, pero se mostrará el menú principal.
- **/Bajar_archivos:** se presentará el menú para la transferencia de archivos.
 - **/Bajar_el_ultimo:** se descargará el último archivo generado.
 - **/Bajar_ultimos_10:** se descargarán los últimos 10 archivos generados en orden del archivo más nuevo al más antiguo.
 - **/Listar_archivos:** se le presentará un listado con los últimos 10 archivos generados orden del más antiguo al más nuevo.
 - **/Anteriores:** envíe este comando para listar los 10 archivos anteriores.
 - **/Posteriores:** envíe este comando para listar los 10 archivos posteriores.
- **/Parametros:** se presentará el menú para la modificación de los parámetros de configuración.
 - **/Mostrar_actuales:** se mostrarán los parámetros de configuración actuales.
 - **/Si:** envíe este comando si desea modificar alguno de los parámetros mostrados.
 - **/No:** envíe este comando si no desea modificar ningún parámetro.
 - **/Modificar_parametros:** se mostrará un listado con las opciones para modificar cada uno de los parámetros.
 - **/Canal_de_disparo:** para seleccionar el canal que provocará la generación de archivos en caso de que alguna muestra supere el nivel de disparo.
 - **/Frecuencia_de_muestreo:** para seleccionar la frecuencia de muestreo del conversor en SPS.
 - **/Nivel_de_disparo:** para enviar el nivel de disparo en volts que provocará la generación de archivos.
 - **/Muestras_post_trigger:** para seleccionar la cantidad de muestras post disparo que se almacenarán.
 - **/Muestras_pre_trigger:** para seleccionar la cantidad de muestras pre disparo que se almacenarán.
 - **/Cantidad_archivos_almacenables:** para seleccionar la cantidad de archivos que se generarán antes de comenzar a sobrescribirlos.
 - **/Aplicar_ahora:** para aplicar los cambios realizados en los parámetros.
 - **/Tal vez_luego:** para continuar sin aplicar los cambios realizados.
- **/Temp:** podrá visualizar la temperatura a la que se encuentra el procesador de la placa Raspberry Pi en grados centígrados.

12 HOJAS DE DATOS

12.1 Raspberry Pi 3 Model B+

Overview



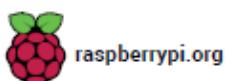
The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT.

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

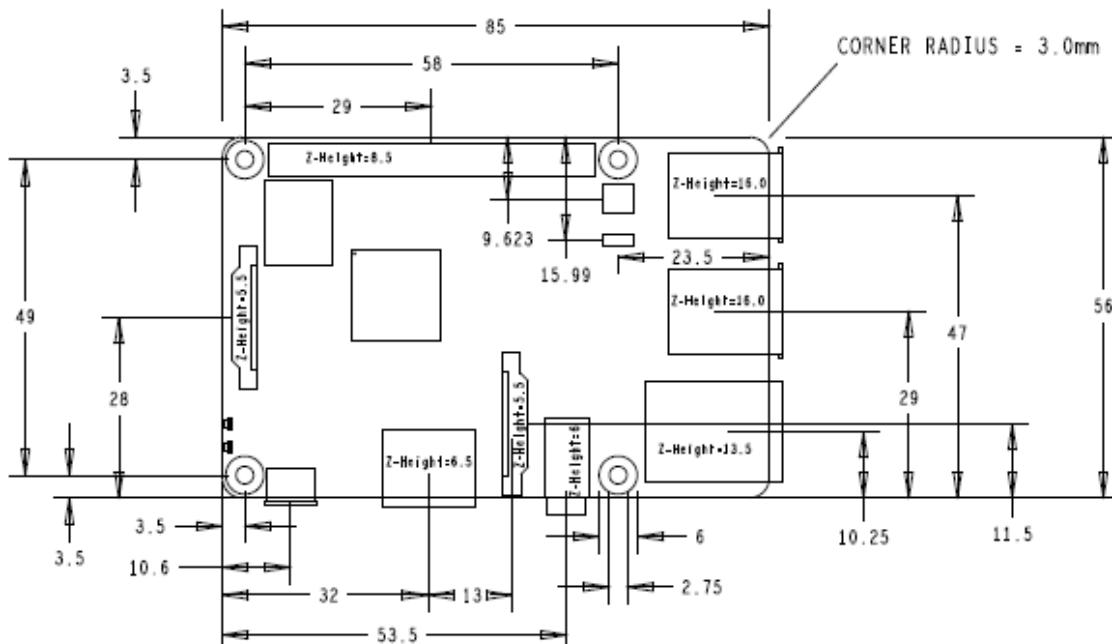
The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	<ul style="list-style-type: none">■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE■ Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	<ul style="list-style-type: none">■ 1 × full size HDMI■ MIPI DSI display port■ MIPI CSI camera port■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	<ul style="list-style-type: none">■ 5V/2.5A DC via micro USB connector■ 5V DC via GPIO header■ Power over Ethernet (PoE)-enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50 °C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5V/2.5A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

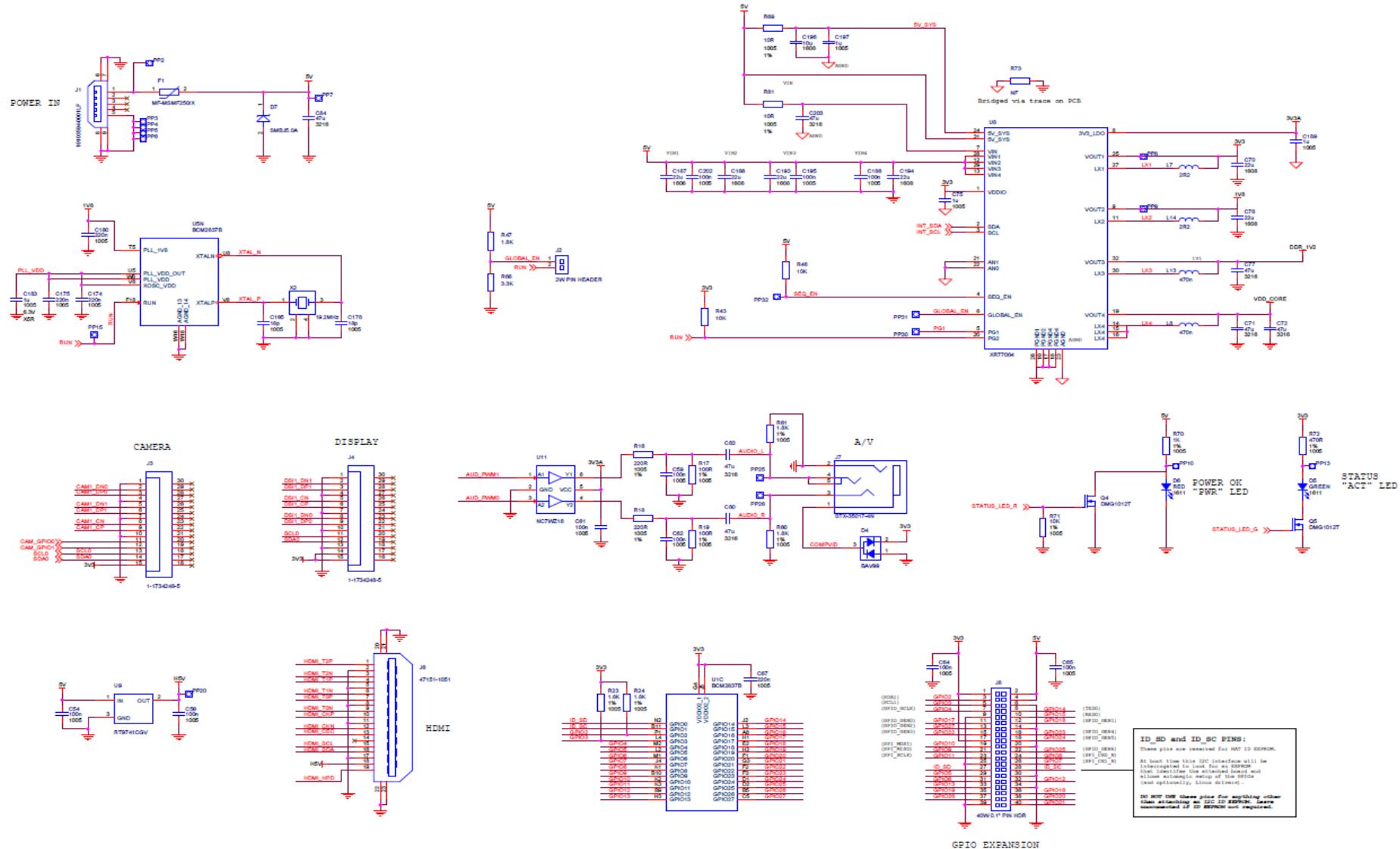
To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.



raspberrypi.org

12.1.1 Circuito esquemático:



12.2 Placa conversora AD-DA

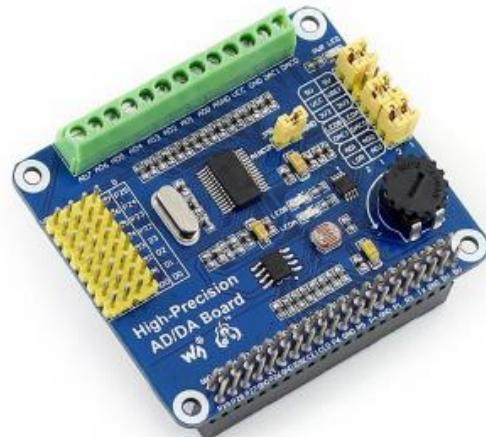
High-Precision AD-DA User Manual



High-Precision AD/DA Board User Manual

Overview

There's no AD/DA function on the Raspberry Pi GPIO interface, this may troubled you in the Pi development. However, it won't be a problem anymore. The High-Precision AD/DA Board allows you to add high-precision AD/DA functions to the Raspberry Pi.



Supported Pi

- Raspberry Pi 1 Model A+
- Raspberry Pi 1 Model B+
- Raspberry Pi 2 Model B

Features

- Onboard ADS1256, 8ch 24bit high-precision ADC (4ch differential input), 30ksps sampling rate
- Onboard DAC8532, 2ch 16bit high-precision DAC
- Onboard input interface via pinheaders, for connecting analog signal
 - the pinout is compatible with Waveshare sensor interface standard, easy to connect various analog sensor modules
- Onboard input/output interface via screw terminals, for connecting analog/digital signal
- Features AD/DA detect circuit, easy for signal demonstration

What's on Board



1. Raspberry Pi GPIO interface: for connecting with the Pi
2. AD/DA input/output: screw terminals
3. AD input: pinheaders, the pinout is compatible with Waveshare sensor interface standard, easy to connect various analog sensor modules
4. 7.68M crystal
5. LM285-2.5: provides reference voltage for the ADC chip
6. Photo resistor
7. LED output indicator
8. 10K potentiometer
9. DAC8532: 16bit high-precision DAC, 2ch
10. Power indicator
11. ADS1256: 24bit high-precision ADC, 8ch (4ch differential input)
12. ADC testing jumper
13. DAC testing jumper
14. Power selection jumper
15. ADC reference ground configuration: when AD single inputted, the AINCOM is reference terminal, can be connected to GND or external reference voltage

Symbol descriptions

1) AD/DA input/output (Tab 2)

AD0-AD7: Analog input

AGND: Analog ground

GND: Digital ground

VCC: Power supply (3.3V and 5V optional, can be switched by setting the Power selection jumper)

DA0-DA1: Analog output

2) Analog input (Tab 3)

AD0-AD7: ADS1256 analog input

D0-D3: GPIO of ADS1256 (See ADS1256 datasheet)

P22-P25: GPIO of Raspberry Pi

AGND: Analog ground

3) LDR: Light-dependent resistor, i.e. Photo resistor (Tab 6)

By connecting the jumper between AD1 and LDR, the MCU can read the voltage output of LDR from the pin AD1.

4) LEDA/LEDB: LED output indicator (Tab 7)

By connecting the jumper between LEDA/LEDB and DAC0/DAC1, the voltage output of DAC0/DAC1 can be estimated roughly by the brightness of LEDA/LEDB.

5) ADJ: The adjustable resistor of the 10K potentiometer (Tab 8)

By connecting the jumper between AD0 and ADJ, the MCU can read the voltage output of the potentiometer from the pin AD0

6) PWR LED: Power indicator (Tab 10)

7) Power selection jumper (Tab 14)

VCC: Power supply selection

VREF: Reference input voltage

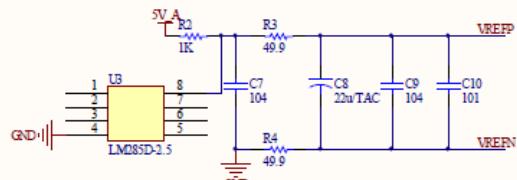
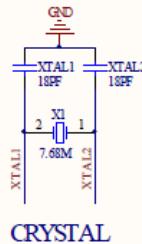
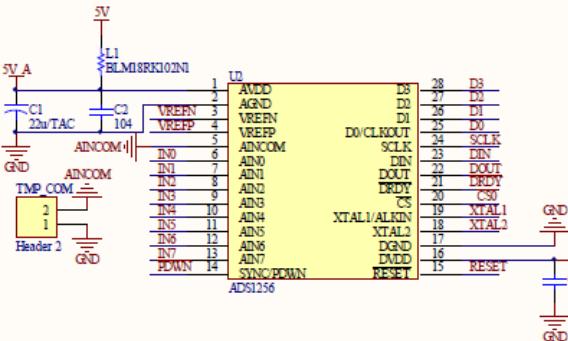
3V3: 3.3V output

5V: 5V output

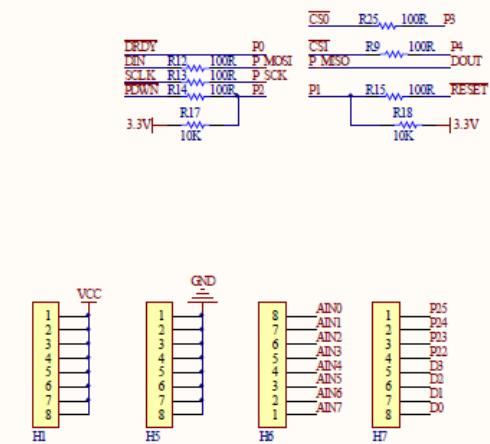
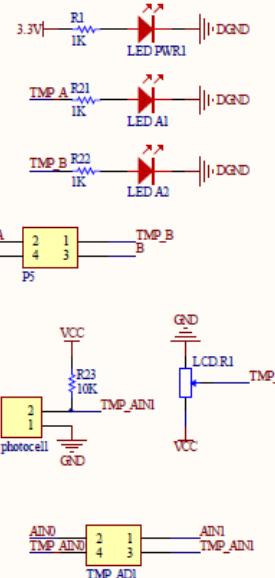
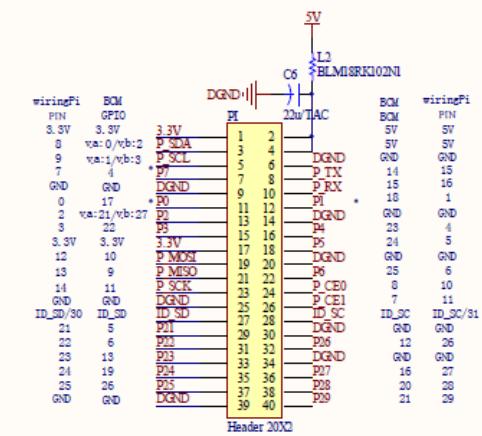
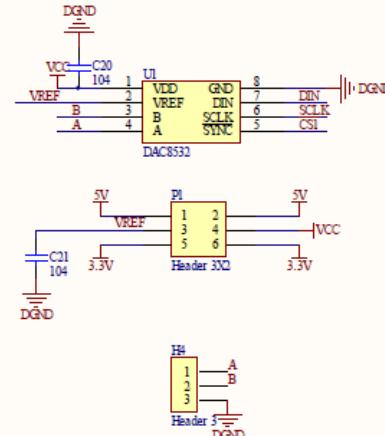
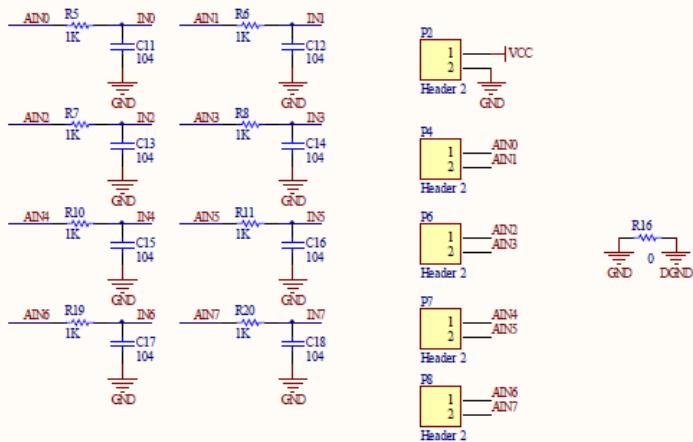
8) JMP_AGND: Analog ground jumper(Tab 15)

For single-ended measurements, use AINCOM (Analog input common) as common input, which can be connected to AGND or external reference voltage. For differential measurements, do not use AINCOM.

12.2.1 Circuito esquemático:



W
Waveform
S
Spot Peak



12.3 Conversor ADS1256

Very Low Noise, 24-Bit Analog-to-Digital Converter

FEATURES

- 24 Bits, No Missing Codes
 - All Data Rates and PGA Settings
- Up to 23 Bits Noise-Free Resolution
- $\pm 0.0010\%$ Nonlinearity (max)
- Data Output Rates to 30kSPS
- Fast Channel Cycling
 - 18.6 Bits Noise-Free (21.3 Effective Bits) at 1.45kHz
- One-Shot Conversions with Single-Cycle Settling
- Flexible Input Multiplexer with Sensor Detect
 - Four Differential Inputs (ADS1256 only)
 - Eight Single-Ended Inputs (ADS1256 only)
- Chopper-Stabilized Input Buffer
- Low-Noise PGA: 27nV Input-Referred Noise
- Self and System Calibration for All PGA Settings
- 5V Tolerant SPI™-Compatible Serial Interface
- Analog Supply: 5V
- Digital Supply: 1.8V to 3.6V
- Power Dissipation
 - As Low as 38mW in Normal Mode
 - 0.4mW in Standby Mode

APPLICATIONS

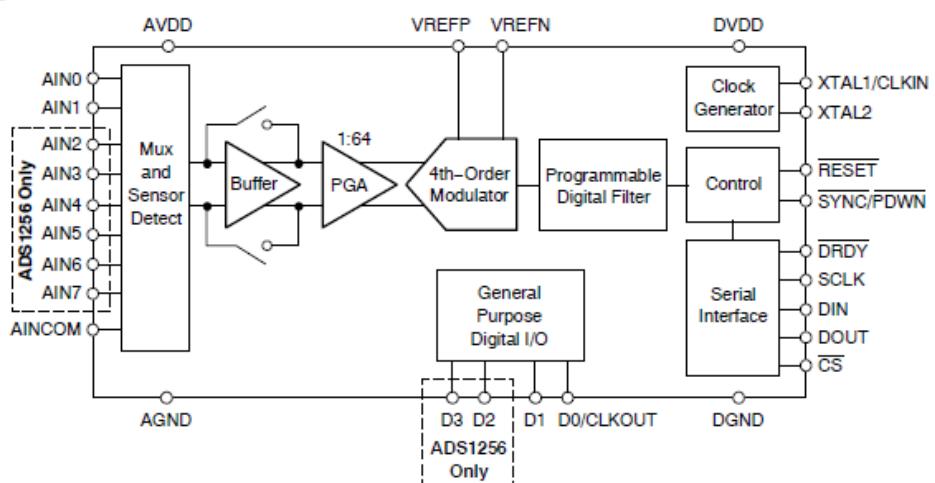
- Scientific Instrumentation
- Industrial Process Control
- Medical Equipment
- Test and Measurement
- Weigh Scales

DESCRIPTION

The ADS1255 and ADS1256 are extremely low-noise, 24-bit analog-to-digital (A/D) converters. They provide complete high-resolution measurement solutions for the most demanding applications.

The converter is comprised of a 4th-order, delta-sigma ($\Delta\Sigma$) modulator followed by a programmable digital filter. A flexible input multiplexer handles differential or single-ended signals and includes circuitry to verify the integrity of the external sensor connected to the inputs. The selectable input buffer greatly increases the input impedance and the low-noise programmable gain amplifier (PGA) provides gains from 1 to 64 in binary steps. The programmable filter allows the user to optimize between a resolution of up to 23 bits noise-free and a data rate of up to 30k samples per second (SPS). The converters offer fast channel cycling for measuring multiplexed inputs and can also perform one-shot conversions that settle in just a single cycle.

Communication is handled over an SPI-compatible serial interface that can operate with a 2-wire connection. Onboard calibration supports both self and system correction of offset and gain errors for all the PGA settings. Bidirectional digital I/Os and a programmable clock output driver are provided for general use. The ADS1255 is packaged in an SSOP-20, and the ADS1256 in an SSOP-28.



ORDERING INFORMATION

For the most current package and ordering information, see the Package Option Addendum at the end of this document, or see the TI web site at www.ti.com.

ABSOLUTE MAXIMUM RATINGS

over operating free-air temperature range unless otherwise noted⁽¹⁾

	ADS1255, ADS1256	UNIT	
AVDD to AGND	-0.3 to +6	V	
DVDD to DGND	-0.3 to +3.6	V	
AGND to DGND	-0.3 to +0.3	V	
Input Current	100, Momentary	mA	
	10, Continuous	mA	
Analog inputs to AGND	-0.3 to AVDD + 0.3	V	
Digital inputs	DIN, SCLK, CS, RESET, SYNC/PDWN, XTAL1/CLKIN to DGND	-0.3 to +6	V
	D0/CLKOUT, D1, D2, D3 to DGND	-0.3 to DVDD + 0.3	V
Maximum Junction Temperature	+150	°C	
Operating Temperature Range	-40 to +105	°C	
Storage Temperature Range	-60 to +150	°C	
Lead Temperature (soldering, 10s)	+300	°C	

⁽¹⁾ Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may degrade device reliability. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those specified is not implied.



This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

ELECTRICAL CHARACTERISTICS

All specifications at -40°C to $+85^{\circ}\text{C}$, AVDD = +5V, DVDD = +1.8V, f_{CLKIN} = 7.68MHz, PGA = 1, and V_{REF} = +2.5V, unless otherwise noted.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Analog Inputs					
Full-scale input voltage (AIN _P – AIN _N)		$\pm 2V_{\text{REF}}/\text{PGA}$			V
Absolute input voltage (AIN0-7, AINCOM to AGND)	Buffer off	AGND – 0.1		AVDD + 0.1	V
	Buffer on	AGND		AVDD – 2.0	V
Programmable gain amplifier		1		64	
Differential input impedance	Buffer off, PGA = 1, 2, 4, 8, 16	150/PGA			kΩ
	Buffer off, PGA = 32, 64	4.7			kΩ
	Buffer on, f _{DATA} ≤ 50Hz ⁽¹⁾	80			MΩ
Sensor detect current sources	SDCS[1:0] = 01	0.5			μA
	SDCS[1:0] = 10	2			μA
	SDCS[1:0] = 11	10			μA
System Performance					
Resolution		24			Bit
No missing codes	All data rates and PGA settings	24			Bit
Data rate (f _{DATA})	f _{CLKIN} = 7.68MHz	2.5		30,000	SPS ⁽²⁾
Integral nonlinearity	Differential input, PGA = 1	±0.0003	±0.0010		%FSR ⁽³⁾
	Differential input, PGA = 64	±0.0007			%FSR
Offset error	After calibration	On the level of the noise			
Offset drift	PGA = 1	±100			nV/°C
	PGA = 64	±4			nV/°C
Gain error	After calibration, PGA = 1, Buffer on	±0.005			%
	After calibration, PGA = 64, Buffer on	±0.03			%
Gain drift	PGA = 1	±0.8			ppm/°C
	PGA = 64	±0.8			ppm/°C
Common-mode rejection	f _{CM} ⁽⁴⁾ = 60Hz, f _{DATA} = 30kSPS ⁽⁵⁾	95	110		dB
Noise		See Noise Performance Tables			
AVDD power-supply rejection	±5% Δ in AVDD	60	70		dB
DVDD power-supply rejection	±10% Δ in DVDD		100		dB
Voltage Reference Inputs					
Reference input voltage (V _{REF})	V _{REF} = V _{REFP} – V _{REFN}	0.5	2.5	2.6	V
Negative reference input (V _{REFN})	Buffer off	AGND – 0.1		V _{REFP} – 0.5	V
	Buffer on ⁽⁶⁾	AGND		V _{REFP} – 0.5	V
Positive reference input (V _{REFP})	Buffer off	V _{REFN} + 0.5		AVDD + 0.1	V
	Buffer on ⁽⁶⁾	V _{REFN} + 0.5		AVDD – 2.0	V
Voltage reference impedance	f _{CLKIN} = 7.68MHz		18.5		kΩ
Digital Input/Output					
V _{IH}	DIN, SCLK, XTAL1/CLKIN, SYNC/PDWN, CS, RESET	0.8 DVDD		5.25	V
	D0/CLKOUT, D1, D2, D3	0.8 DVDD		DVDD	V
V _{IL}		DGND		0.2 DVDD	V
V _{OH}	I _{OH} = 5mA	0.8 DVDD			V
V _{OL}	I _{OL} = 5mA			0.2 DVDD	V
Input hysteresis			0.5		V
Input leakage	0 < V _{DIGITAL INPUT} < DVDD			±10	μA
Master clock rate	External crystal between XTAL1 and XTAL2	2	7.68	10	MHz
	External oscillator driving CLKIN	0.1	7.68	10	MHz

ELECTRICAL CHARACTERISTICS (continued)

All specifications at -40°C to $+85^{\circ}\text{C}$, $\text{AVDD} = +5\text{V}$, $\text{DVDD} = +1.8\text{V}$, $f_{\text{CLKIN}} = 7.68\text{MHz}$, $\text{PGA} = 1$, and $V_{\text{REF}} = +2.5\text{V}$, unless otherwise noted.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Power-Supply					
AVDD		4.75	5.25		V
DVDD		1.8	3.6		V
AVDD current	Power-down mode		2		μA
	Standby mode		20		μA
	Normal mode, $\text{PGA} = 1$, Buffer off	7	10		mA
	Normal mode, $\text{PGA} = 64$, Buffer off	16	22		mA
	Normal mode, $\text{PGA} = 1$, Buffer on	13	19		mA
	Normal mode, $\text{PGA} = 64$, Buffer on	36	50		mA
DVDD current	Power-down mode		2		μA
	Standby mode, CLKOUT off, $\text{DVDD} = 3.3\text{V}$		95		μA
	Normal mode, CLKOUT off, $\text{DVDD} = 3.3\text{V}$	0.9	2		mA
Power dissipation	Normal mode, $\text{PGA} = 1$, Buffer off, $\text{DVDD} = 3.3\text{V}$	38	57		mW
	Standby mode, $\text{DVDD} = 3.3\text{V}$	0.4			mW
Temperature Range					
Specified		-40		+85	$^{\circ}\text{C}$
Operating		-40		+105	$^{\circ}\text{C}$
Storage		-60		+150	$^{\circ}\text{C}$

(1) See text for more information on input impedance.

(2) SPS = samples per second.

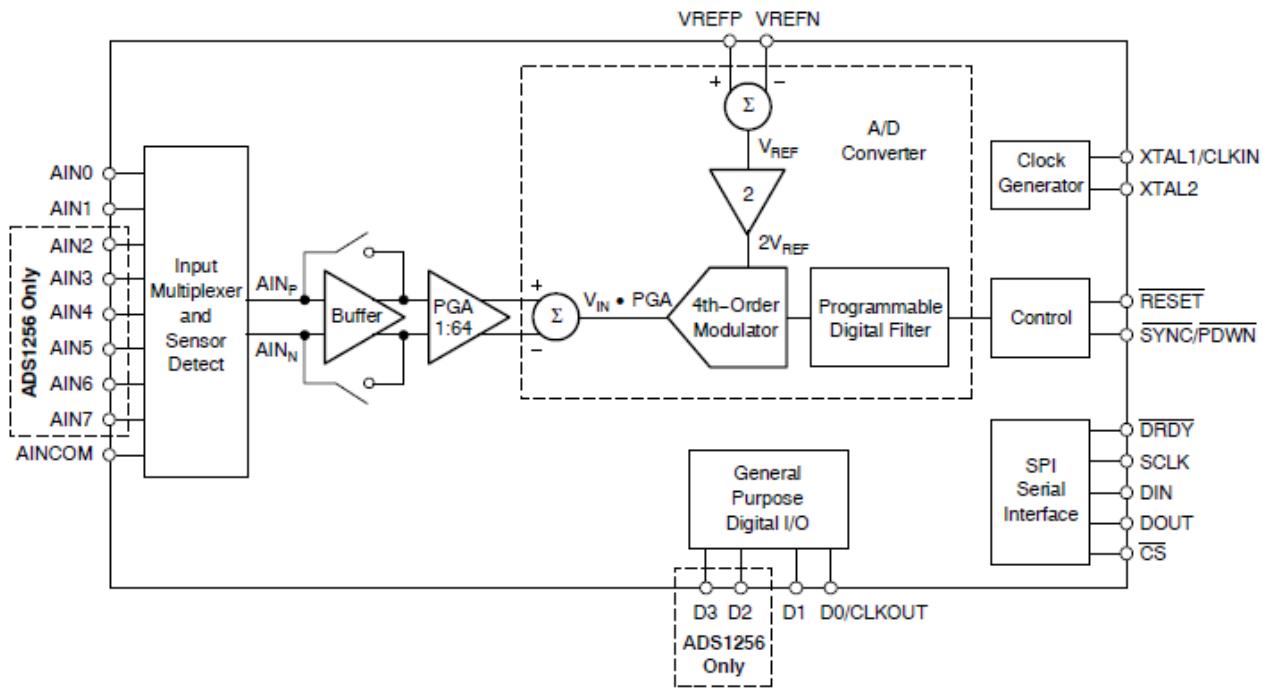
(3) FSR = full-scale range = $4V_{\text{REF}}/\text{PGA}$.

(4) f_{CM} is the frequency of the common-mode input signal.

(5) Placing a notch of the digital filter at 60Hz (setting $f_{\text{DATA}} = 60\text{SPS}$, 30SPS, 15SPS, 10SPS, 5SPS, or 2.5SPS) will further improve the common-mode rejection of this frequency.

(6) The reference input range with Buffer on is restricted only if self-calibration or gain self-calibration is to be used. If using system calibration or writing calibration values directly to the registers, the entire Buffer off range can be used.

12.3.1 Diagrama en bloques:



12.4 Módulo RTC DS3231

DS3231

Extremely Accurate I²C-Integrated
RTC/TCXO/Crystal

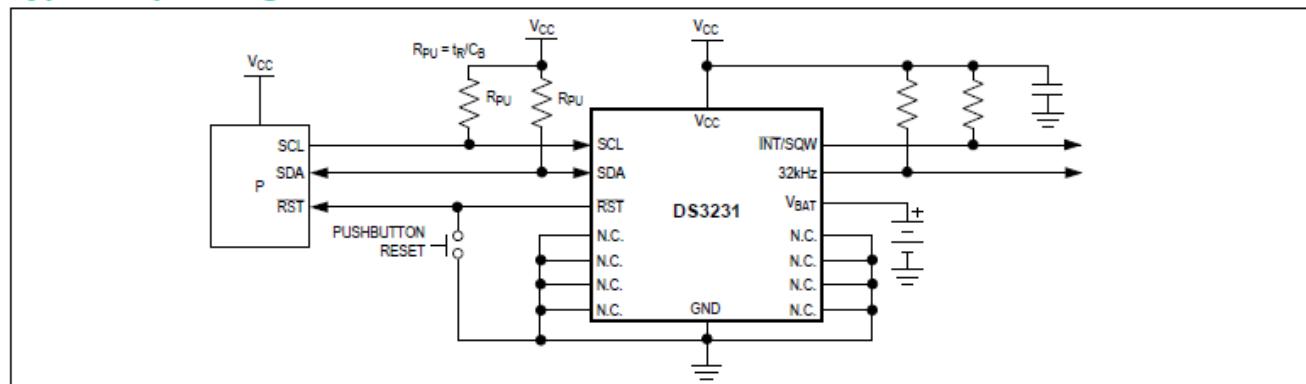
General Description

The DS3231 is a low-cost, extremely accurate I²C real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package.

The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I²C bidirectional bus.

A precision temperature-compensated voltage reference and comparator circuit monitors the status of V_{CC} to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the RST pin is monitored as a pushbutton input for generating a µP reset.

Typical Operating Circuit



Underwriters Laboratories is a registered certification mark of Underwriters Laboratories Inc.

Benefits and Features

- Highly Accurate RTC Completely Manages All Timekeeping Functions
 - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
 - Accuracy ±2ppm from 0°C to +40°C
 - Accuracy ±3.5ppm from -40°C to +85°C
 - Digital Temp Sensor Output: ±3°C Accuracy
 - Register for Aging Trim
 - RST Output/Pushbutton Reset Debounce Input
 - Two Time-of-Day Alarms
 - Programmable Square-Wave Output Signal
- Simple Serial Interface Connects to Most Microcontrollers
 - Fast (400kHz) I²C Interface
- Battery-Backup Input for Continuous Timekeeping
 - Low Power Operation Extends Battery-Backup Run Time
 - 3.3V Operation
- Operating Temperature Ranges: Commercial (0°C to +70°C) and Industrial (-40°C to +85°C)
- Underwriters Laboratories® (UL) Recognized

Applications

- Servers
- Telematics
- Utility Power Meters
- GPS

Ordering Information and Pin Configuration appear at end of data sheet.

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground	-0.3V to +6.0V	Junction Temperature	+125°C
Junction-to-Ambient Thermal Resistance (θ_{JA}) (Note 1)	73°C/W	Storage Temperature Range	-40°C to +85°C
Junction-to-Case Thermal Resistance (θ_{JC}) (Note 1)	23°C/W	Lead Temperature (soldering, 10s)	+260°C
Operating Temperature Range		Soldering Temperature (reflow, 2 times max)	+260°C
DS3231S	0°C to +70°C	(see the <i>Handling, PCB Layout, and Assembly</i> section)	
DS3231SN.....	-40°C to +85°C		

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Recommended Operating Conditions

($T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	V
Logic 1 Input SDA, SCL	V_{IH}		0.7 x V_{CC}	$V_{CC} + 0.3$		V
Logic 0 Input SDA, SCL	V_{IL}		-0.3	0.3 x V_{CC}		V

Electrical Characteristics

($V_{CC} = 2.3V$ to $5.5V$, V_{CC} = Active Supply (see Table 1), $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted.) (Typical values are at $V_{CC} = 3.3V$, $V_{BAT} = 3.0V$, and $T_A = +25^\circ C$, unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS	
Active Supply Current	I_{CCA}	(Notes 4, 5)	$V_{CC} = 3.63V$		200	μA	
			$V_{CC} = 5.5V$		300		
Standby Supply Current	I_{CCS}	I ² C bus inactive, 32kHz output on, SQW output off (Note 5)	$V_{CC} = 3.63V$		110	μA	
			$V_{CC} = 5.5V$		170		
Temperature Conversion Current	$I_{CCSConv}$	I ² C bus inactive, 32kHz output on, SQW output off	$V_{CC} = 3.63V$		575	μA	
			$V_{CC} = 5.5V$		650		
Power-Fail Voltage	V_{PF}			2.45	2.575	2.70	V
Logic 0 Output, 32kHz, INT/SQW, SDA	V_{OL}	$I_{OL} = 3mA$			0.4		V
Logic 0 Output, RST	V_{OL}	$I_{OL} = 1mA$			0.4		V
Output Leakage Current 32kHz, INT/SQW, SDA	I_{LO}	Output high impedance	-1	0	+1	μA	
Input Leakage SCL	I_{LI}		-1		+1	μA	
RST Pin I/O Leakage	I_{OL}	RST high impedance (Note 6)	-200		+10	μA	
V_{BAT} Leakage Current (V_{CC} Active)	I_{BATLKG}			25	100	nA	

Electrical Characteristics (continued)

(V_{CC} = 2.3V to 5.5V, V_{CC} = Active Supply (see Table 1), T_A = T_{MIN} to T_{MAX} , unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 2, 3)

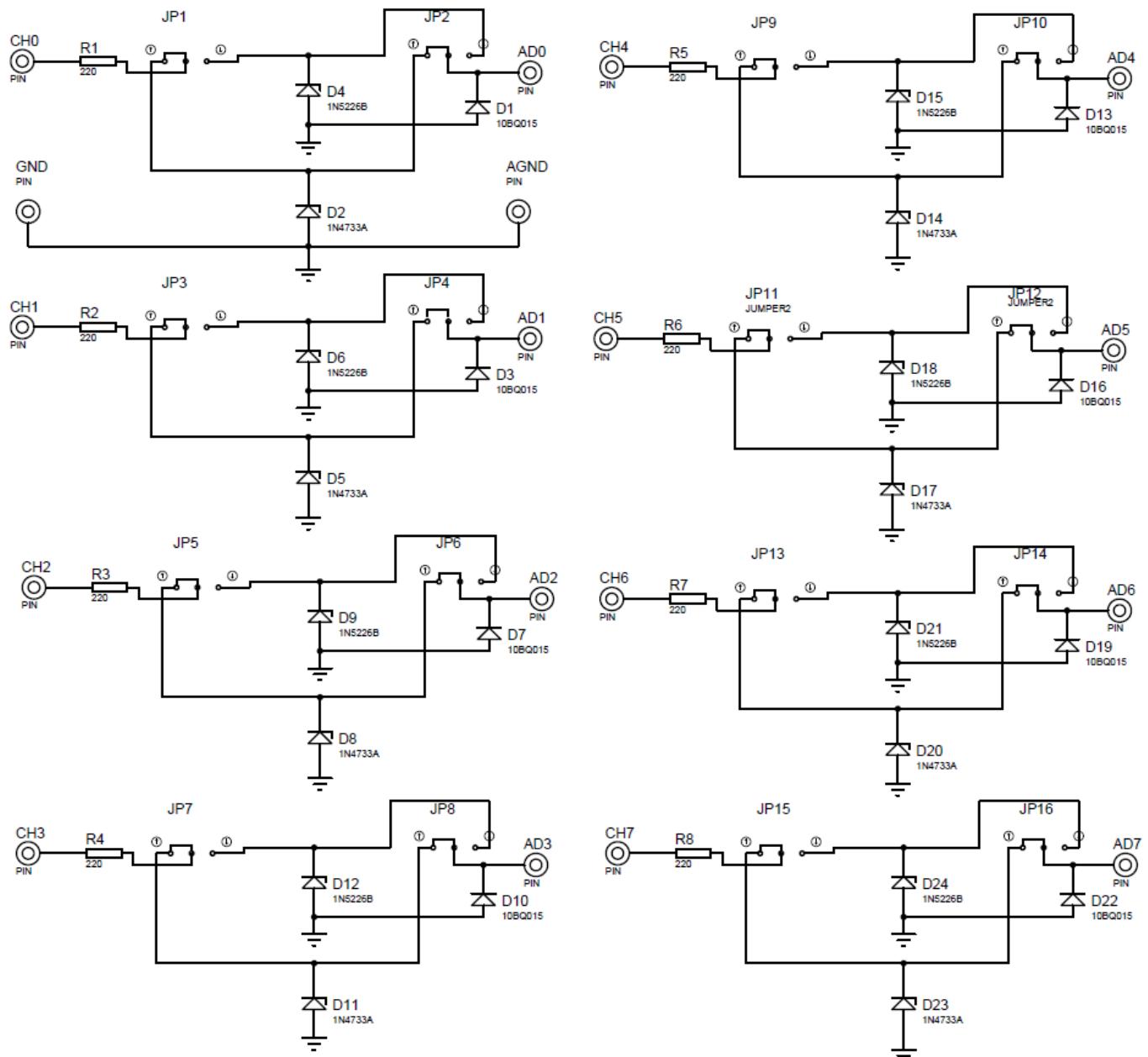
PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS		
Output Frequency	f_{OUT}	V_{CC} = 3.3V or V_{BAT} = 3.3V		32.768		kHz			
Frequency Stability vs. Temperature (Commercial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	0°C to +40°C	± 2		± 3.5	ppm		
			>40°C to +70°C	± 3.5					
Frequency Stability vs. Temperature (Industrial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	-40°C to <0°C	± 3.5		± 2	ppm		
			0°C to +40°C	± 2					
			>40°C to +85°C	± 3.5					
Frequency Stability vs. Voltage	$\Delta f/V$			1		ppm/V			
Trim Register Frequency Sensitivity per LSB	$\Delta f/LSB$	Specified at:	-40°C	0.7		± 0.1	ppm		
			+25°C	0.1					
			+70°C	0.4					
			+85°C	0.8					
Temperature Accuracy	Temp	V_{CC} = 3.3V or V_{BAT} = 3.3V		-3	+3		°C		
Crystal Aging	$\Delta f/f_O$	After reflow, not production tested	First year	± 1.0		± 5.0	ppm		
			0–10 years	± 5.0					

Electrical Characteristics

(V_{CC} = 0V, V_{BAT} = 2.3V to 5.5V, T_A = T_{MIN} to T_{MAX} , unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Battery Current	I_{BATA}	$EOSC = 0$, $BBSQW = 0$, $SCL = 400\text{kHz}$ (Note 5)	$V_{BAT} = 3.63\text{V}$	70		μA	
			$V_{BAT} = 5.5\text{V}$	150			
Timekeeping Battery Current	I_{BATT}	$EOSC = 0$, $BBSQW = 0$, $EN32\text{kHz} = 1$, $SCL = SDA = 0\text{V}$ or $SCL = SDA = V_{BAT}$ (Note 5)	$V_{BAT} = 3.63\text{V}$	0.84		3.0	μA
			$V_{BAT} = 5.5\text{V}$	1.0		3.5	
Temperature Conversion Current	I_{BATTC}	$EOSC = 0$, $BBSQW = 0$, $SCL = SDA = 0\text{V}$ or $SCL = SDA = V_{BAT}$	$V_{BAT} = 3.63\text{V}$	575		μA	
			$V_{BAT} = 5.5\text{V}$	650			
Data-Retention Current	I_{BATTDR}	$EOSC = 1$, $SCL = SDA = 0\text{V}$, +25°C		100		nA	

12.5 Circuito de protección



12.5.1 Diodo Zener 1N5226B



www.vishay.com

1N5221B to 1N5267B

Vishay Semiconductors

Small Signal Zener Diodes



FEATURES

- Silicon planar power Zener diodes
- Standard Zener voltage tolerance is $\pm 5\%$
- These diodes are also available in MiniMELF case with the type designation TZM5221 to TZM5267, SOT-23 case with the type designations MMBZ5225 to MMBZ5267 and SOD-123 case with the types designations MMSZ5225 to MMSZ5267
- AEC-Q101 qualified
- Compliant to RoHS Directive 2002/95/EC and in accordance to WEEE 2002/96/EC
- Halogen-free according to IEC 61249-2-21 definition



RoHS
COMPLIANT

HALOGEN
FREE

PRIMARY CHARACTERISTICS		
PARAMETER	VALUE	UNIT
V _Z range nom.	2.4 to 75	V
Test current I _{ZT}	1.7 to 20	mA
V _Z specification	Thermal equilibrium	
Int. construction	Single	

APPLICATIONS

- Voltage stabilization

ORDERING INFORMATION			
DEVICE NAME	ORDERING CODE	TAPED UNITS PER REEL	MINIMUM ORDER QUANTITY
1N5221B to 1N5267B	1N5221B to 1N5267B-series-TR	10 000 per 13" reel	30 000/box
1N5221B to 1N5267B	1N5221B to 1N5267B-series-TAP	10 000 per ammopack (52 mm tape)	30 000/box

PACKAGE				
PACKAGE NAME	WEIGHT	MOLDING COMPOUND FLAMMABILITY RATING	MOISTURE SENSITIVITY LEVEL	SOLDERING CONDITIONS
DO-35	125 mg	UL 94 V-0	MSL level 1 (according J-STD-020)	260 °C/10 s at terminals

ABSOLUTE MAXIMUM RATINGS ($T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified)				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Power dissipation	$T_L \leq 25\text{ }^{\circ}\text{C}$	P _{tot}	500	mW
Zener current		I _Z	P _{tot} /V _Z	mA
Thermal resistance junction to ambient air	I = 4 mm, T_L = constant	R _{thJA}	300	K/W
Junction temperature		T _j	175	°C
Storage temperature range		T _{stg}	- 65 to + 175	°C
Forward voltage (max.)	I _F = 200 mA	V _F	1.1	V

ELECTRICAL CHARACTERISTICS ($T_{amb} = 25^\circ C$, unless otherwise specified)							
PART NUMBER	ZENER VOLTAGE RANGE ⁽¹⁾	TEST CURRENT		REVERSE LEAKAGE CURRENT		DYNAMIC RESISTANCE $f = 1 \text{ kHz}$	
	Vz at IzT1	IzT1	IzT2	Ir at VR	Zz at IzT1 ⁽¹⁾	Zzk at IzT2	αV
	V	mA		μA	V	Ω	
	NOM.			MAX.		MAX.	TYP.
1N5221B	2.4	20	0.25	100	1	30	1200
1N5222B	2.5	20	0.25	100	1	30	1250
1N5223B	2.7	20	0.25	75	1	30	1300
1N5224B	2.8	20	0.25	75	1	30	1400
1N5225B	3	20	0.25	50	1	29	1600
1N5226B	3.3	20	0.25	25	1	28	1600
1N5227B	3.6	20	0.25	15	1	24	1700
1N5228B	3.9	20	0.25	10	1	23	1900
1N5229B	4.3	20	0.25	5	1	22	2000
1N5230B	4.7	20	0.25	5	2	19	1900
1N5231B	5.1	20	0.25	5	2	17	1600
1N5232B	5.6	20	0.25	5	3	11	1600
1N5233B	6	20	0.25	5	3.5	7	1600
1N5234B	6.2	20	0.25	5	4	7	1000
1N5235B	6.8	20	0.25	3	5	5	750
1N5236B	7.5	20	0.25	3	6	6	500
1N5237B	8.2	20	0.25	3	6.5	8	500
1N5238B	8.7	20	0.25	3	6.5	8	600
1N5239B	9.1	20	0.25	3	7	10	600
1N5240B	10	20	0.25	3	8	17	600
1N5241B	11	20	0.25	2	8.4	22	600
1N5242B	12	20	0.25	1	9.1	30	600
1N5243B	13	9.5	0.25	0.5	9.9	13	600
1N5244B	14	9	0.25	0.1	10	15	600
1N5245B	15	8.5	0.25	0.1	11	16	600
1N5246B	16	7.8	0.25	0.1	12	17	600
1N5247B	17	7.4	0.25	0.1	13	19	600
1N5248B	18	7	0.25	0.1	14	21	600
1N5249B	19	6.6	0.25	0.1	14	23	600
1N5250B	20	6.2	0.25	0.1	15	25	600
1N5251B	22	5.6	0.25	0.1	17	29	600
1N5252B	24	5.2	0.25	0.1	18	33	600
1N5253B	25	5	0.25	0.1	19	35	600
1N5254B	27	4.6	0.25	0.1	21	41	600
1N5255B	28	4.5	0.25	0.1	21	44	600
1N5256B	30	4.2	0.25	0.1	23	49	600
1N5257B	33	3.8	0.25	0.1	25	58	700
1N5258B	36	3.4	0.25	0.1	27	70	700
1N5259B	39	3.2	0.25	0.1	30	80	800
1N5260B	43	3	0.25	0.1	33	93	900
1N5261B	47	2.7	0.25	0.1	36	105	1000
1N5262B	51	2.5	0.25	0.1	39	125	1100
1N5263B	56	2.2	0.25	0.1	43	150	1300
1N5264B	60	2.1	0.25	0.1	46	170	1400
1N5265B	62	2	0.25	0.1	47	185	1400
1N5266B	68	1.8	0.25	0.1	52	230	1600
1N5267B	75	1.7	0.25	0.1	56	270	1700

Note

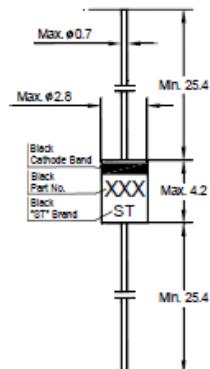
(1) Based on DC measurement at thermal equilibrium; lead length = 9.5 (3/8"); thermal resistance of heat sink = 30 K/W

12.5.2 Diodo Zener 1N4733A

1N4727A...1N4764A

SILICON PLANAR POWER ZENER DIODES

for use in stabilizing and clipping circuits with high power rating.



Glass Case DO-41
Dimensions in mm

Absolute Maximum Ratings ($T_a = 25^\circ\text{C}$)

Parameter	Symbol	Value	Unit
Power Dissipation	P_{tot}	1 ¹⁾	W
Junction Temperature	T_j	200	°C
Storage Temperature Range	T_s	- 65 to + 200	°C

¹⁾ Valid provided that leads at a distance of 8 mm from case are kept at ambient temperature.

Characteristics at $T_a = 25^\circ\text{C}$

Parameter	Symbol	Max.	Unit
Thermal Resistance Junction to Ambient Air	R_{thA}	170 ¹⁾	K/W
Forward Voltage at $I_F = 200 \text{ mA}$	V_F	1.2	V

¹⁾ Valid provided that leads at a distance of 8 mm from case are kept at ambient temperature.

Type	Zener Voltage Range ^{3), 5)}			Maximum Zener Impedance ¹⁾			Maximum Reverse Leakage Current		Maximum Surge Current ⁴⁾	Maximum Regulator Current ²⁾
	V _{Znom}	V _Z	I _{ZT}	r _{ZJT}	r _{ZJK}	at I _{ZK}	I _R	at V _R	at T _a = 25 °C	
	V	V	mA	Ω	Ω	mA	μA	V	I _{ZSM} (mA)	I _{ZM} (mA)
1N4727A	3	2.85...3.15	83	10	400	1	150	1	1375	275
1N4728A	3.3	3.13...3.47	76	10	400	1	150	1	1375	275
1N4729A	3.6	3.42...3.78	69	10	400	1	100	1	1260	252
1N4730A	3.9	3.7...4.1	64	9	400	1	100	1	1190	234
1N4731A	4.3	4.08...4.52	58	9	400	1	50	1	1070	217
1N4732A	4.7	4.46...4.94	53	8	500	1	10	1	970	193
1N4733A	5.1	4.84...5.36	49	7	550	1	10	1	890	178
1N4734A	5.6	5.32...5.88	45	5	600	1	10	2	810	162
1N4735A	6.2	5.89...6.51	41	2	700	1	10	3	730	146
1N4736A	6.8	6.46...7.14	37	3.5	700	1	10	4	660	133
1N4737A	7.5	7.12...7.88	34	4	700	0.5	10	5	605	121
1N4738A	8.2	7.79...8.61	31	4.5	700	0.5	10	6	550	110
1N4739A	9.1	8.64...9.56	28	5	700	0.5	10	7	500	100
1N4740A	10	9.5...10.5	25	7	700	0.25	10	7.6	454	91
1N4741A	11	10.45...11.55	23	8	700	0.25	5	8.4	414	83
1N4742A	12	11.4...12.6	21	9	700	0.25	5	9.1	380	76
1N4743A	13	12.35...13.65	19	10	700	0.25	5	9.9	344	69
1N4744A	15	14.25...15.75	17	14	700	0.25	5	11.4	304	61
1N4745A	16	15.2...16.8	15.5	16	700	0.25	5	12.2	285	57
1N4746A	18	17.1...18.9	14	20	750	0.25	5	13.7	250	50
1N4747A	20	19...21	12.5	22	750	0.25	5	15.2	225	45
1N4748A	22	20.9...23.1	11.5	23	750	0.25	5	16.7	205	41
1N4749A	24	22.8...25.2	10.5	25	750	0.25	5	18.2	190	38
1N4750A	27	25.65...28.35	9.5	35	750	0.25	5	20.6	170	34
1N4751A	30	28.5...31.5	8.5	40	1000	0.25	5	22.8	150	30
1N4752A	33	31.35...34.65	7.5	45	1000	0.25	5	25.1	135	27
1N4753A	36	34.2...37.8	7	50	1000	0.25	5	27.4	125	25
1N4754A	39	37.05...40.95	6.5	60	1000	0.25	5	29.7	115	23
1N4755A	43	40.85...45.15	6	70	1500	0.25	5	32.7	110	22
1N4756A	47	44.65...49.35	5.5	80	1500	0.25	5	35.8	95	19
1N4757A	51	48.45...53.55	5	95	1500	0.25	5	38.8	90	18
1N4758A	56	53.2...58.8	4.5	110	2000	0.25	5	42.6	80	16
1N4759A	62	58.9...65.1	4	125	2000	0.25	5	47.1	70	14
1N4760A	68	64.6...71.4	3.7	150	2000	0.25	5	51.7	65	13
1N4761A	75	71.25...78.75	3.3	175	2000	0.25	5	56	60	12
1N4762A	82	77.9...86.1	3	200	3000	0.25	5	62.2	55	11
1N4763A	91	86.45...95.55	2.8	250	3000	0.25	5	69.2	50	10
1N4764A	100	95...105	2.5	350	3000	0.25	5	76	45	9

¹⁾ The Zener Impedance is derived from the 60 Hz AC voltage which results when an AC current having an RMS value equal to 10% of the Zener Current (I_{ZT} or I_{ZK}) is superimposed on I_{ZT} or I_{ZK}. Zener Impedance is measured at two points to insure a sharp knee on the breakdown curve and to eliminate unstable units.

²⁾ Valid provided that leads at a distance of 8 mm from case are kept at ambient temperature.

³⁾ Measured under thermal equilibrium and DC test conditions.

⁴⁾ The rating listed in the electrical characteristics table is maximum peak, non-repetitive, reverse surge current of 1/2 square wave or equivalent sine wave pulse of 1/120 second duration superimposed on the test current I_{ZT}.

⁵⁾ Tested with pulses t_p = 20 ms.

12.5.3 Diodo Schottky 10BQ015



PD - 2.396A

10BQ015

SCHOTTKY RECTIFIER

1 Amp

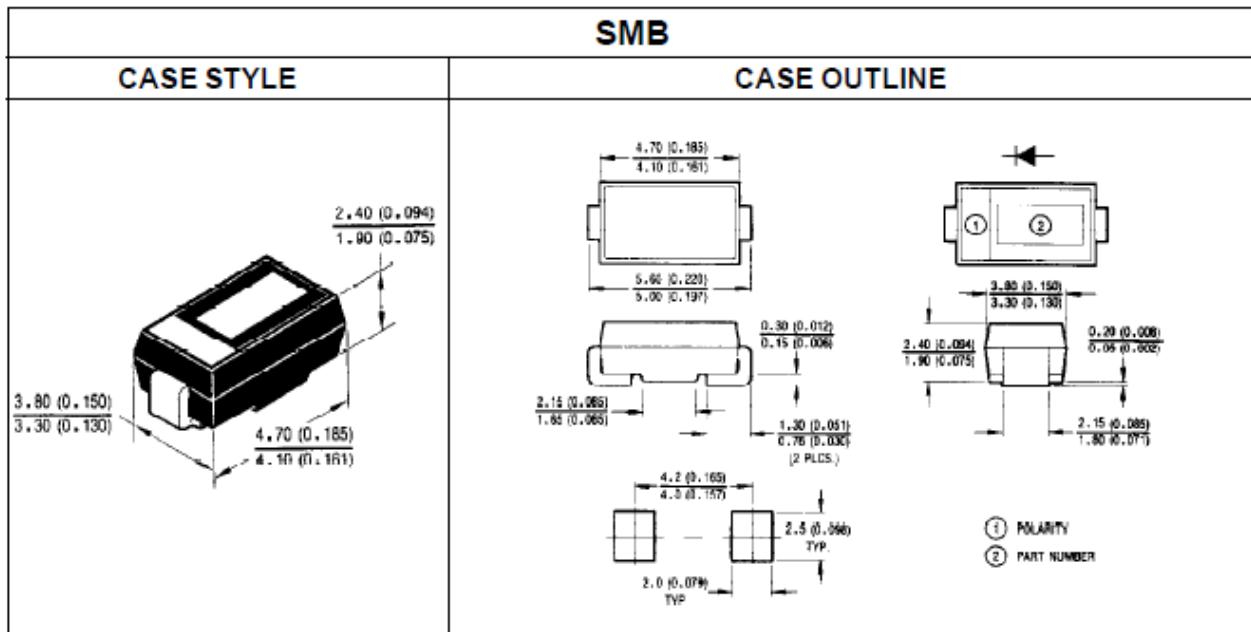
Major Ratings and Characteristics

Characteristics	10BQ015	Units
I _{F(AV)} Rectangular waveform	1.0	A
V _{RRM}	15	V
I _{FSM} @ t _p = 5μs sine	140	A
V _F @ 1.0Apk, T _J = 75°C	0.30	V
T _J	-55 to 100	°C

Description / Features

The 10BQ015 surface-mount Schottky rectifier has been designed for applications requiring very low forward drop and small foot prints on PC boards. Typical applications are in disk drives, switching power supplies, converters, free-wheeling diodes, battery charging and reverse battery protection.

- Small footprint, surface mountable
- Very low forward voltage drop
- High frequency operation
- Guard ring for enhanced ruggedness and long-term reliability



Voltage Ratings

Part number		10BQ015
V_R	Max. DC Reverse Voltage (V)	15
V_{RWM}	Max. Working Peak Reverse Voltage (V)	25

Absolute Maximum Ratings

Parameters		10BQ	Units	Conditions
$I_{F(AV)}$	Max. Average Forward Current See Fig. 5	1.0	A	50% duty cycle @ $T_c = 78^\circ\text{C}$, rectangular waveform
I_{FSM}	Max. Peak One Cycle Non - Repetitive Surge Current — see Fig. 7	140	A	5μs Sine or 3μs Rect. pulse
		40		10ms Sine Or 6ms Rect. pulse
E_{AS}		5.0	mJ	$T_J = 25^\circ\text{C}$, $I_{AS} = 0.2\text{A}$, $L = 4.2\text{mH}$
I_{AR}	Repetitive Avalanche Current	0.2	A	Current decaying linearly to zero in 1μsec Frequency limited by T_J max. $V_A = 1.5 \times V_R$ typical

Electrical Specifications

Parameters		10BQ	Units		Conditions
V_{FM}	Max. Forward Voltage Drop See Fig. 1 ①	0.34	V	@ 1.0A	$T_J = 25^\circ\text{C}$
		0.40	V	@ 2.0A	
		0.30	V	@ 1.0A	$T_J = 75^\circ\text{C}$
		0.38	V	@ 2.0A	
I_{RM}	Max. Reverse Leakage Current ② See Fig. 2	0.50	mA	$T_J = 25^\circ\text{C}$	$V_R = \text{rated } V_R$
		12	mA	$T_J = 100^\circ\text{C}$	
C_T	Max. Junction Capacitance	390	pF	$V_R = 5\text{V}_{\text{DC}}$, (test signal range 100KHz to 1MHz)	25°C
L_s	Typical Series Inductance	2.0	nH	Measured lead to lead 5mm from package body	
dv/dt	Max. Voltage Rate of Change (Rated V_R)	6,000	V/μs		

Thermal-Mechanical Specifications

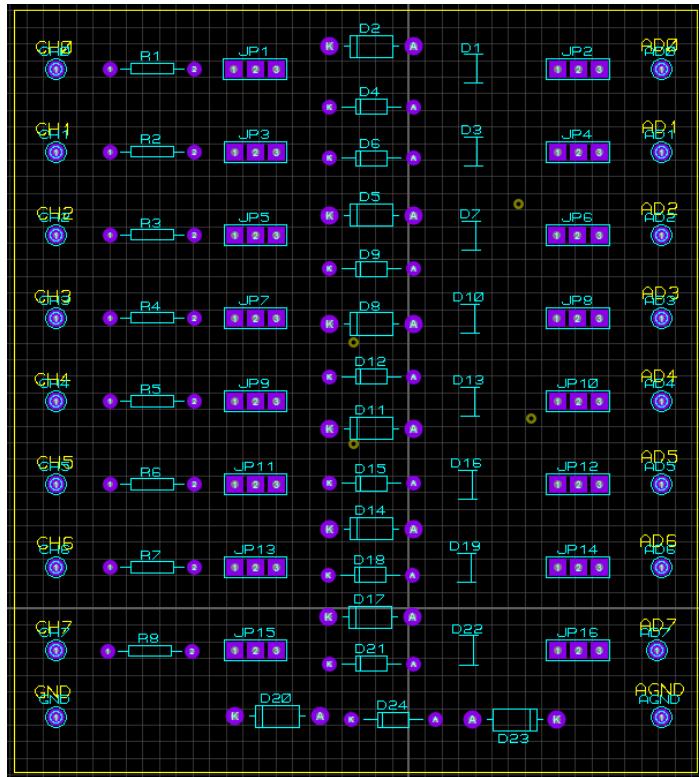
Parameters		10BQ	Units	Conditions
T_J	Max.Junction Temperature Range	-55 to 100	°C	
T_{STG}	Max. Storage Temperature Range	-55 to 100	°C	
R_{thJA}	Max. Thermal Resistance, Junction to Ambient	140	°C/W	DC operation — See Fig. 4
R_{thJL}	Max. Thermal Resistance, Junction to Lead ③	36	°C/W	DC operation
wt	Approximate Weight	0.10	g	
Case Style		SMB	Similar to DO-214AA	

① Pulse Width < 300μs, Duty Cycle < 2%

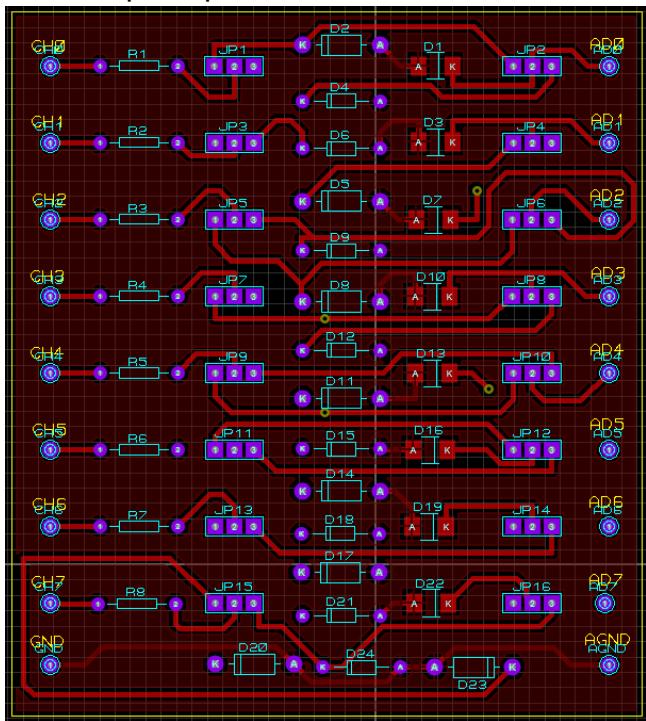
② Mounted 1 inch square PCB, thermal probe connected to lead 2mm from package

12.5.4 Diseño PCB

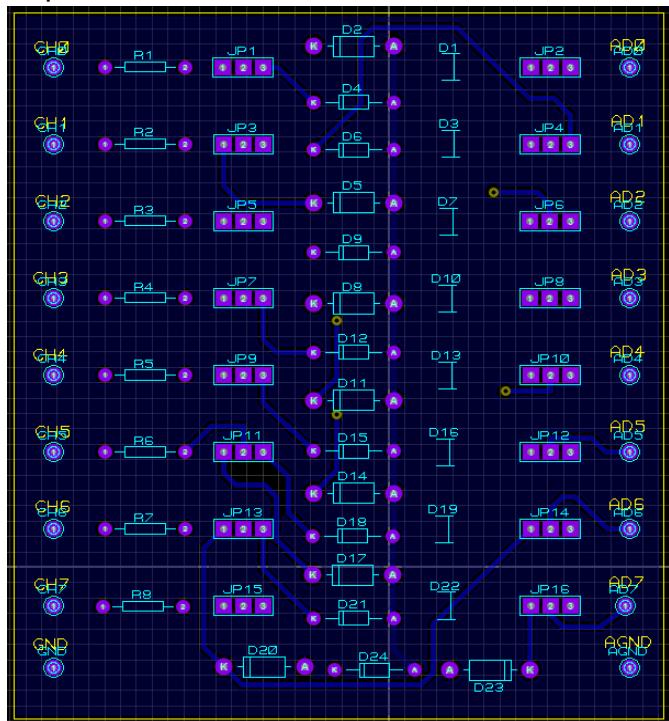
Componentes:



Capa superior:



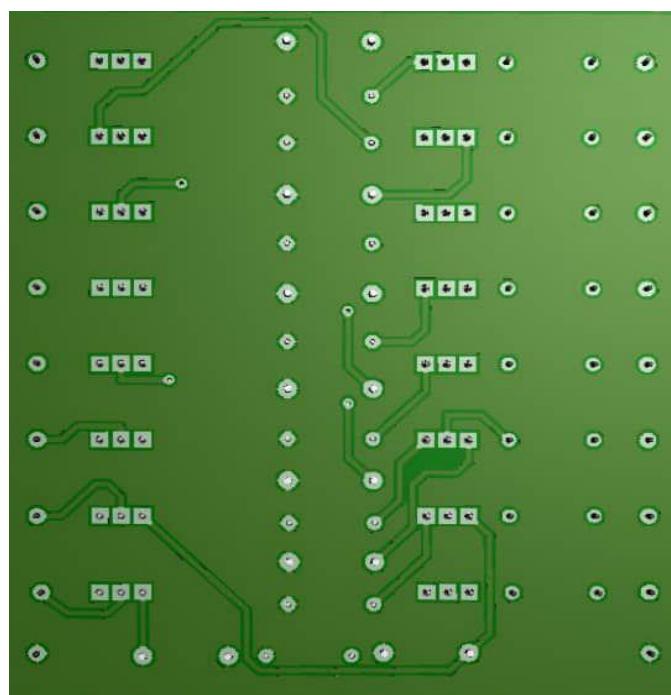
Capa inferior:



Vista 3D:
Superior:



Inferior:



13 BIBLIOGRAFÍA

- Adafruit.* (s.f.). Obtenido de Set RTC Time: <https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-rtc-time>
- Airspayce.* (s.f.). Recuperado el 5 de Abril de 2022, de C library for Broadcom BCM 2835 as used in Raspberry Pi: <http://www.airspayce.com/mikem/bcm2835/>
- Baúl de Linux.* (s.f.). Obtenido de <https://baulderasec.wordpress.com/apuntes/linux/4-configurar-los-discos-duros/4-9-herramientas-para-encontrar-ficheros/4-9-1-el-comando-find/>
- Chicala, I. C. (2004). *Adquisición De Datos: Medir para Conocer y Controlar.* Soluciones en Control S.R.L.
- Flopy.* (21 de 07 de 2019). Obtenido de Crea un bot de Telegram para tu Raspberry. Ordénale cosas y habla con ella a distancia.: <https://www.flopy.es/crea-un-bot-de-telegram-para-tu-raspberry-ordenale-cosas-y-habla-con-ella-a-distancia/>
- GitHub.* (s.f.). Obtenido de Telegram Bot API: <https://github.com/tdlib/telegram-bot-api>
- GitHub.* (25 de Diciembre de 2021). Obtenido de Telebot: <https://github.com/smartnode/telebot>
- GitHub.* (2019 de Enero de 22). Obtenido de RaspberryPi-ADC-DAC: <https://github.com/AKEOPLUS-boris-bocquet/RaspberryPi-ADC-DAC>
- g-se.* (24 de Enero de 2013). Obtenido de Medida ± Desviación Estándar o ± Error Estándar de la Media: <https://g-se.com/medida-desviacion-estandar-o-error-estandar-de-la-media-bp-k57cfb26ceb5e5>
- Hostinger.* (7 de Diciembre de 2021). Obtenido de Cómo usar los comandos Find y Locate en Linux: <https://www.hostinger.es/tutoriales/como-usar-comando-find-locate-en-linux/>
- Hostinger.* (29 de Julio de 2021). Obtenido de Cómo utilizar SFTP (Protocolo de transferencia de archivos SSH): <https://www.hostinger.com.ar/tutoriales/como-usar-sftp>
- Hwlible.* (s.f.). Recuperado el Octubre de 2021, de GPIO: todo sobre las conexiones de la Raspberry Pi 4 y 3: <https://www.hwlible.com/gpio-raspberry-pi/>
- Keithley, J. F. (s.f.). *Low Level Measurements* (5th Edition ed.).
- kolwidi.* (15 de Abril de 2021). Obtenido de MÓDULO RTC Y RASPBERRY PI PARA CONTROLAR EL TIEMPO: <https://kolwidi.com/blogs/blog-kolwidi/modulo-rtc-ds1307-y-raspberry-pi>

Linuxito. (9 de Noviembre de 2012). Obtenido de Cómo buscar archivos por nombre: <https://www.linuxito.com/gnu-linux/nivel-basico/114-como-buscar-archivos-por-nombre>

Malvino, A. P. (2000). *PRINCIPIOS DE ELECTRÓNICA* (Sexta ed.). McGraw-Hill.

Mercado Libre. (5 de Abril de 2022). Obtenido de Ds3231 Rtc De Alta Precisión Raspberry Pi: https://articulo.mercadolibre.com.ar/MLA-724733357-ds3231-rtc-de-alta-precision-raspberry-pi-_JM?matt_tool=88481412&matt_word=&matt_source=google&matt_campaign_id=11618987428&matt_ad_group_id=113657532672&matt_match_type=&matt_network=g&matt_device=c&matt

Monkiki. (22 de Marzo de 2017). Obtenido de RTC DS3231 en Raspberry Pi: <https://monkiki.github.io/2017/03/22/rtc-ds3231-en-raspberry-pi.html>

Poesiabinaria. (7 de Diciembre de 2017). Obtenido de Crea rápidamente servicios con Systemd para iniciar demonios o ejecutar scripts: <https://poesiabinaria.net/2017/12/crea-rapidamente-servicios-systemd/>

Rapipc. (27 de Junio de 2015). Obtenido de Añadir un Reloj RTC a la Raspberry Pi: <http://www.raspipc.es/blog/anadir-un-reloj-rtc-a-la-raspberry-pi/>

Redeweb. (1 de Febrero de 2021). Obtenido de Conexión de Telegram con Raspberry Pi Chateando con nuestros dispositivos: <https://www.redeweb.com/articulos/conexion-de-telegram-con-raspberry-pi/>

Stackoverflow. (18 de 8 de 2016). Obtenido de Create a file if one doesn't exist - C: <https://stackoverflow.com/questions/9840629/create-a-file-if-one-doesnt-exist-c>

Telegram. (s.f.). Recuperado el Noviembre de 2021, de Telegram Bot API: <https://core.telegram.org/bots/api>

Telegram. (s.f.). Obtenido de Bots: An introduction for developers: <https://core.telegram.org/bots>

Tutorialspoint. (s.f.). Obtenido de C library function - strftime(): https://www.tutorialspoint.com/c_standard_library/c_function_strerror.htm

Tutorialspoint. (s.f.). Recuperado el Agosto de 2021, de C library function - fopen(): https://www.tutorialspoint.com/c_standard_library/c_function_fopen.htm

Voragine. (26 de Dic de 2016). Obtenido de CÓMO CREAR Y GESTIONAR DEMONIOS EN LINUX CON SYSTEMD: <https://voragine.net/linux/como-crear-y-gestionar-demonios-en-linux-con-systemd>

Waveshare. (s.f.). Recuperado el 10 de Agosto de 2021, de High-Precision AD/DA Board: https://www.waveshare.com/wiki/High-Precision_AD/DA_Board