UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UN MEDIDOR TRIFÁSICO MULTIFUNCIÓN CON MONITOREO ARMÓNICO UTILIZANDO EL IC ADE7880

PRESENTADO POR:

FÉLIX ENRIQUE CAMPOS MARROQUÍN

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, FEBRERO DE 2022

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL:

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

PHD. EDGAR ARMANDO PEÑA FIGUEROA

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR:

ING.ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título:

DISEÑO E IMPLEMENTACIÓN DE UN MEDIDOR TRIFÁSICO MULTIFUNCIÓN CON MONITOREO ARMÓNICO UTILIZANDO EL IC ADE7880

Presentado por:

FÉLIX ENRIQUE CAMPOS MARROQUÍN

Trabajo de Graduación Aprobado por:

Docente Asesor:

ING. HUGO MIGUEL COLATO RODRÍGUEZ

SAN SALVADOR, FEBRERO DE 2022

Trabajo de Graduación Aprobado por:

Docente Asesor:

ING. HUGO MIGUEL COLATO RODRÍGUEZ

NOTA Y DEFENSA FINAL

En esta fecha, Jueves 3 de febrero de 2022, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 5:00 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director

Firma

2. MSc. José Wilber Calderón Urrutia
   Secretario

Por: Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- MSC. HUGO MIGUEL COLATO RODRIGUEZ
  (Docente Asesor)

Firma

- DR. CARLOS OSMIN POCASANGRE JIMENEZ

Firma

- MSC. JOSÉ WILBER CALDERÓN URRUTIA

Por: Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

DISEÑO E IMPLEMENTACIÓN DE UN MEDIDOR TRIFÁSICO MULTIFUNCIÓN CON MONITOREO ARMÓNICO, UTILIZANDO EL IC ADE7880

A cargo del Bachiller:

- FÉLIX ENRIQUE CAMPOS MARROQUÍN

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 8.6

(ocho punto seis)

AGRADECIMIENTOS

A mis padres Ana de Campos y Agustín Campos que siempre me apoyaron para poder concluir mi formación profesional, también les agradezco todas las buenas costumbres que me han enseñado a lo largo de mi vida.

A mis hermanos Juan Campos y Arquímedes Campos por su apoyo y animarme siempre a salir adelante.

A mi primo Giovanni Hasmed Rodríguez Marroquín con quien compartí gran parte de mi vida, vivencias, aspiraciones y muchas alegrías. Que descanses en paz.

A mis sobrinos Ismael Campos y Hasmed Campos.

A mis tías y tíos que me dieron su apoyo, consejos, cariño y comprensión, recuerdos invaluables que guardare hasta el último momento de mi existencia.

A todos mis amigos y compañeros con los que compartí esta gran experiencia de nuestra formación profesional, en especial a los que estuvimos en la asociación de ingeniería eléctrica de esa época.

Agradezco a todas aquellas personas que de alguna forma compartieron con migo múltiples experiencias porque estas han contribuido en mí.

Agradezco al personal de la Universidad de El Salvador y especialmente al docente, administrativo y de laboratorio de la carrera de ingeniería eléctrica que me guiaron por el sendero del conocimiento, a mi asesor de tesis Ing. Hugo Colato por guiarme e impulsarme a concluir este trabajo.

FÉLIX ENRIQUE CAMPOS MARROQUÍN

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

SIGET	Súper Intendencia General de Electricidad y Telecomunicaciones.
CI	Circuito Integrado.
DSP	Procesador de Señales Digital.
PGA	Amplificadores de Ganancia Programable.
MEMORIA ROM	Memoria de solo Lectura.
MEMORIA RAM	Memoria de datos volátil.
I2C	Circuito inter-integrado. Es un protocolo de comunicación serial.
SBC	Single Board Computer. Es un ordenador completo de placa única quiere decir que la memoria RAM los puertos E/S el microprocesador y demás componentes se encuentran incluidos en una sola PCB.
RTC	Real-Time Clock. Es un reloj calendario en tiempo real.
ACK	Acknowledgement. Acuse de recibido o asentimiento. Es un tipo de mensaje que se envía en un tipo de comunicación serial.
NACK	No Acknowledgement.
CODIGO FUENTE	Es un programa informático o conjunto de líneas de código con los pasos que debe ejecutar el microprocesador/microcontrolador.
IDE	Integrated Development Environment. Es un entorno de desarrollo integrado que proporciona servicios integrales para facilitarle al programador el desarrollo de software.
S.O	Sistema Operativo.
JVM	Java Virtual Machine. Es una máquina virtual de java y su función es interpretar y ejecutar el código bytecode el cual es generado por el compilador java.
POO	Programación Orientada a Objetos.
SQL	Structured Query Languaje. Es un lenguaje de consulta estructurada. Diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.
HTTP	Hiper Text Transfer Protocol. Es el protocolo de comunicación que permite las transferencias de información través de archivos HTML en el internet.
JSP	Java Server Page. Es una tecnología utilizada para crear páginas web dinámicas.
GPIO	General Purpose Input Output. Es un puerto de entrada y salida de propósito general, consta de una serie de pines que se pueden configurar como entradas o salidas digitales.
METROLOGÍA	Es la ciencia que tiene por objeto el estudio de los sistemas de medida, la determinación de magnitudes físicas y la aplicación de los medios o procesos para la medición de las magnitudes.

CALIBRACIÓN	La calibración es el procedimiento metrológico por medio del cual se compara un equipo de medición cualquiera con un equipo patrón de referencia cuya exactitud es determinante para dicho proceso
PATRÓN DE MEDIDA	Es una referencia que se utiliza para realizar la calibración de los equipos, el patrón de medida puede ser un instrumento de medida, una medida materializada o un sistema de medida. El patrón de referencia tiene la más alta calidad metrológica disponible en un laboratorio del cual se derivan las mediciones efectuadas en dicho establecimiento.
AJUSTE	Es la operación de llevar un instrumento de medición a un correcto funcionamiento, en el cual la exactitud, precisión e incertidumbre están dentro de los parámetros de tolerancia para su utilización.
TRAZABILIDAD	La trazabilidad es una propiedad del proceso de medición donde el resultado puede ser relacionado por medio de una cadena ininterrumpida de comparaciones entre patrones nacionales e internacionales teniendo todas las incertidumbres determinadas.

INTRODUCCIÓN

Los medidores de energía eléctrica o Watthorímetros electromecánicos fueron de las primeras tecnologías que se utilizaron para medir el consumo de energía eléctrica, tanto en los hogares como en la industria. Los equipos de medición han mejorado con el paso del tiempo. Con los avances tecnológicos y científicos han logrado reducir el tamaño de los dispositivos o sustituirlos por otros mejorando sus prestaciones y rendimiento. Hoy en día se utilizan en su gran mayoría equipos de medición electrónicos con los cuales se mejora la calidad de las mediciones y proporcionan mucha más información útil, tanto en el consumo de la energía eléctrica así como en los procesos de generación, transmisión y distribución de energía eléctrica. Con la finalidad de evaluar la calidad de la energía que llega al usuario final.

En el primer capítulo se revisan los conceptos fundamentales implicados en la medición de energía eléctrica así como también sus expresiones matemáticas. Además se describen las características más relevantes de los dispositivos electrónicos, una mini computadora raspberry pi, el protocolo de comunicación I2C, el puerto de propósito general GPIO. Además se describen las herramientas de software utilizados en el desarrollo de este trabajo de grado. Tales como el software CAD para el diseño del modelo 3D de la carcasa del equipo de medición, los IDEs como entorno para el desarrollo del software, el diseño de interfaces gráficas y bases de datos para el desarrollo del software de control. También un software para el diseño de circuitos impresos y finalmente el servidor de aplicaciones web que se utiliza en el diseño de la solución del equipo de medición.

En el capítulo dos se presenta el diseño del hardware y sus distintos componentes que se modelan en diagramas de bloques los cuales describen el funcionamiento de las partes que conforman el equipo de medición, así como la interconexión de los dispositivos eléctricos y electrónicos, selección de valores de componentes resistivos y capacitivos, configuración, calibración y puesta en marcha del medidor trifásico.

En el capítulo tres se presenta el desarrollo del software que tiene la función de soportar el hardware, se parte de un diagrama en bloques del funcionamiento del gestor interno y un diagrama de modelo cliente/servidor. Además se presenta la configuración e integración de las partes que conforman el sistema con el servidor de aplicaciones WEB.

OBJETIVOS

OBJETIVO GENERAL

Diseñar y desarrollar el prototipo de un medidor trifásico de energía eléctrica con monitoreo armónico utilizando como base el circuito integrado ADE7880.

OBJETIVOS ESPECÍFICOS

- Crear el prototipo de un medidor trifásico de energía eléctrica en el cual se almacenen mediciones de parámetros eléctricos tales como corrientes, voltajes, potencias, factor de potencia y distorsión armónica total para la corriente y el voltaje.

- Hacer uso del protocolo de comunicaciones I2C para la obtención de datos por medio de una placa raspberry pi, utilizando esta última como infraestructura servidor de base de datos y servidor de publicación web.

- Desarrollar una aplicación cliente/servidor en lenguaje de programación java para el procesamiento y presentación de reportes.

- Desarrollar una aplicación móvil en la plataforma Android para acceder a reportes del equipo de forma remota.

CAPÍTULO I

MARCO TEÓRICO

1.0 Introducción

En este primer capítulo se describen los conceptos fundamentales que dan soporte al proceso de medición de energía eléctrica, se abordan las tecnologías existentes. Además se presenta el software y librerías seleccionados para el diseño y desarrollo de software tanto para el lado del servidor así como para el lado del cliente. Se describen las características más importantes de los componentes electrónicos que se utilizaran en el desarrollo como por ejemplo un reloj de tiempo real, así como el medidor dedicado ADE7880 de la empresa analog devices y una placa raspberry pi 4. La teoría de operación del puerto de comunicaciones I2C y del puerto de propósito general GPIO. También se presentan algunos entornos de desarrollo o IDEs para la programación de software que dará soporte al hardware.

1.1 Medición de energía eléctrica

La medición de la energía eléctrica consiste en cuantificar la potencia consumida por una carga en un intervalo de tiempo. La potencia es el resultado de medir la magnitud de voltaje rms, la magnitud de corriente rms y en sistemas de corriente alterna el factor de potencia (Fp), con estos tres parámetros se puede definir matemáticamente la potencia eléctrica promedio en un sistema de corriente alterna.

1.2 Medición de corriente eléctrica

Para llevar a cabo la medición de corriente eléctrica se hace uso de transductores para poder convertir esta magnitud física en una señal eléctrica útil, existen una gran variedad de tecnologías que proporcionan una solución. Entre las que podemos mencionar están: los sensores de efecto hall, los transformadores de corriente, los sensores de corriente shunt y las bobinas de rogowski. Para el presente trabajo de grado se hace uso de la tecnología de los transformadores de corriente de núcleo partido.

1.2.1 Transformadores de corriente eléctrica (TC)

Los transformadores son máquinas eléctricas destinadas a reducir y en algunos casos a aumentar la magnitud física del nivel de voltaje o del nivel de intensidad de corriente. Estas máquinas están compuestas por dos devanados uno llamado primario y el segundo llamado secundario los cuales tienen N_1 espiras y N_2 espiras respectivamente, el circuito magnético de esta máquina lo conforma un núcleo magnético echo con chapas de acero al silicio individuales generalmente para reducir las pérdidas. La operación del transformador de corriente se describe a continuación. En el devanado primario únicamente tiene una espira la cual es el conductor de interés este debe atravesar por el centro al núcleo y debido al campo magnético que la corriente que circula por este conductor genera, induce una corriente en el devanado secundario con una magnitud menor pero proporcional a la magnitud del devanado

primario, esta relación entre la corriente en el devanado primario y el devanado secundario recibe el nombre de relación de transformación y se puede expresar matemáticamente así:

$$\frac{N_p}{N_s} = \frac{I_s}{I_p}$$ **Ecuación 1.0**

Donde:
N_P = Numero de vueltas o espiras en el devanado primario (conductor de interés).
N_S = Numero de vueltas o espiras en el devanado secundario.
I_S = Corriente inducida en el devanado secundario.
I_P = Corriente en el devanado primario.
En la figura 1 se muestra el funcionamiento y el esquema eléctrico de un TC

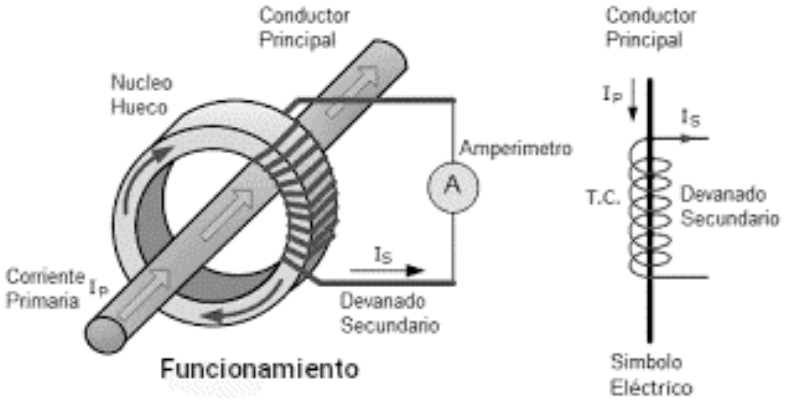


Figura 1: Transformador de corriente.

1.3 Medición de voltaje

Para llevar a cabo la medición del voltaje se necesita hacer uso de transformadores de voltaje en algunos diseños podemos prescindir de ellos y en su lugar implementar un circuito llamado divisor de tensión o incluso utilizar ambos, esta elección entre una solución con o sin transformador depende del diseño electrónico. En este caso lo importante es obtener una magnitud de voltaje adecuada para poder ser procesada en un microcontrolador o como es en este caso en un circuito integrado medidor dedicado ADE7880.

1.3.1 Transformador de voltaje

Estos transformadores al igual que con los transformadores de corriente son máquinas eléctricas, su funcionamiento es un poco distinto al TC, pero los principios y las leyes físicas que los rigen son las mismas, la relación de transformación para un transformador de voltaje se expresa a continuación: $$\frac{N_p}{N_s} = \frac{V_p}{V_s}$$ **Ecuación 1.1**

En la figura 2 se muestra el esquema eléctrico de un transformador de voltaje. En este trabajo de grado no se hará uso de los transformadores de voltaje, en su lugar se utilizara el circuito divisor de voltaje resistivo.

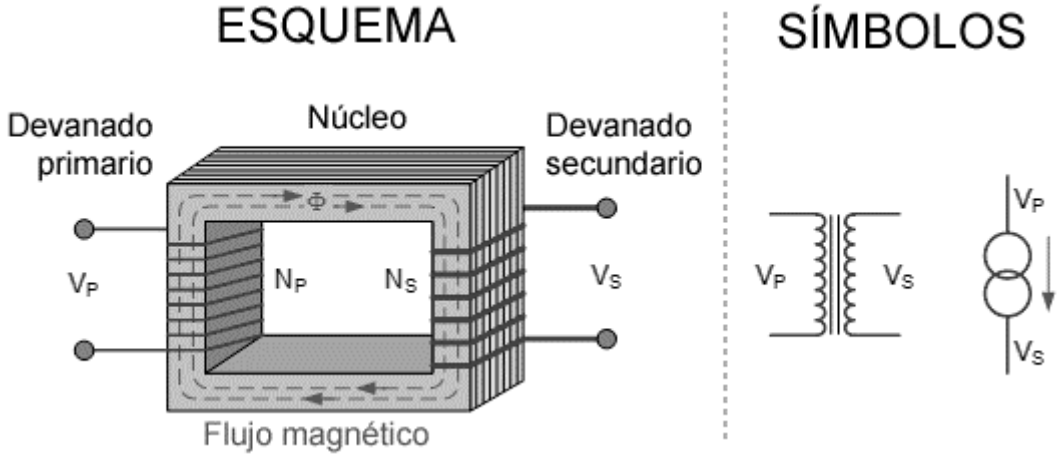


Figura 2: Transformador de voltaje.

1.4 Potencia activa

En un circuito eléctrico en corriente alterna la potencia activa es la que desarrollar trabajo útil. Y puede valerse para ello de una conversión en otras formas energéticas tales como: calor, luz, sonido, movimiento y en procesos electro químicos. Su unidad en el sistema internacional de unidades es el vatio y su símbolo es "W". Para un sistema en corriente alterna la expresión matemática que describe a la potencia activa es la siguiente:

$$\boldsymbol{Potencia_{ACTIVA} = V_{rms} * I_{rms} * cos\,\theta}$$ **Ecuación 1.2**

1.5 Potencia reactiva

Es la potencia que se genera cuando existen cargas reactivas (bobinas y/o condensadores) en un circuito eléctrico, esta potencia no desarrolla trabajo útil sin embargo es requerida por las cargas reactivas para poder generar campos magnéticos y campos eléctricos, necesarios para el buen funcionamiento de las diferentes cargas conectadas a una fuente de corriente alterna. El símbolo que la representa es "Q" y su unidad es voltio-amperio reactivo o VAR. Su representación matemática es la siguiente: $\boldsymbol{Q = V_{rms} * I_{rms} * \sin\theta}$ **Ecuación 1.3**

1.6 Potencia aparente

Esta potencia, viene dada por la relación del producto del voltaje y la corriente eficaces respectivamente o también de extraer el módulo de la potencia compleja[1], esta magnitud es útil para determinar la relación del factor de potencia. Esta potencia muestra la demanda total que requiera una carga o un conjunto de ellas puesto que incluye a la potencia activa y la

potencia reactiva. El símbolo que representa a la potencia aparente es “S” y su unidad es voltio-amperio o VA. Su representación matemática se muestra a continuación:

$$|\boldsymbol{S}| = \boldsymbol{V_{rms}} * \boldsymbol{I_{rms}} = \sqrt{\boldsymbol{P^2 + Q^2}} \quad \textbf{Ecuación 1.4}$$

1.7 Factor de potencia (FP)

Este parámetro eléctrico surge de la relación entre la potencia activa y la potencia aparente, por lo que proporciona una medida de la eficiencia con la que se desarrolla trabajo útil en un sistema eléctrico en corriente alterna. Es una magnitud adimensional su expresión matemática es: $FP = \frac{Potencia\ Activa}{Potencia\ Aparente} = \frac{P}{|S|}$ $Ecuación\ 1.5$ Este parámetro es muy importante para las empresas distribuidoras de energía eléctrica ya que está bajo norma por la SIGET. El rango permitido en el que deben operar los sistemas eléctricos industriales debe estar entre 0.90 hasta 1.0, siendo el límite inferior el de mayor rigor, si cae por debajo de este valor estará sujeto a penalizaciones económicas según la normativa vigente.

1.7 Distorsión armónica

Son señales de voltajes y corrientes senoidales presentes en un sistema eléctrico, cuya frecuencia son múltiplos enteros de la frecuencia de la señal fundamental, las amplitudes de estas señales armónicas pueden alterar el valor eficaz ocasionando un mal funcionamiento de los equipos conectados en un circuito eléctrico. La distorsión armónica es causada debido a cargas no lineales conectadas al sistema eléctrico de corriente alterna y para su análisis se hace uso generalmente de las series y transformadas de fourier[2].

1.7.1 Distorsión armónica total (THD)

El THD es uno de los índices utilizados para analizar los efectos producidos por los armónicos, el cual cuantifica el nivel de contaminación armónica de las señales de voltaje y corriente en un sistema eléctrico. Este índice se define matemáticamente como sigue:

$$\boldsymbol{para\ el\ voltaje} \quad \boldsymbol{THD_V} = \frac{\sqrt{\sum_{i=2}^{\infty} \boldsymbol{V_i}^2}}{\boldsymbol{V_1}} * \boldsymbol{100\%} \quad \textbf{Ecuación 1.6}$$

$$\boldsymbol{para\ la\ corriente} \quad \boldsymbol{THD_I} = \frac{\sqrt{\sum_{i=2}^{\infty} \boldsymbol{I_i}^2}}{\boldsymbol{I_1}} * \boldsymbol{100\%} \quad \textbf{Ecuación 1.7}$$

1 La potencia compleja S = P + jQ.
2 Jean-Batiste Josep Fourier [Auxerre (Francia) 1768 – París 1830] Matemático y Físico Francés.

1.8 Medidores de energía eléctrica

Los medidores de energía eléctrica son los instrumentos dedicados para medir diferentes Parámetros eléctricos en un punto de conexión en la red eléctrica, estos parámetros se utilizan para cuantificar el consumo de energía eléctrica, a la vez que proporcionan información muy importante que se utiliza para su análisis y optimización, con la finalidad de tener un sistema eléctrico eficiente. Existen diferentes tecnologías de medición las cuales han evolucionado al ritmo de los avances tecnológicos, en general se pueden clasificar estas tecnologías en tres tipos las cuales son:

- Medidores electromecánicos.
- Medidores híbridos.
- Medidores electrónicos (de mayor uso en la actualidad).

1.8.1 Medidor de energía electromecánico

Los medidores de energía eléctrica o watthorimetros electromecánicos fueron de las primeras tecnologías que se utilizaron para solucionar el problema de medir el consumo de la energía eléctrica, en la figura 3 se muestra un equipo de medición electromecánico. El uso de esta tecnología en la actualidad ha decaído significativamente. Se pueden mencionar las partes que conforman un medidor de energía eléctrica electromecánico, como se detalla a continuación en la figura 4:

Figura 3: Medidor electromecánico.

1. Bobina de voltaje.
2. Bobina de corriente.
3. imán de frenado.
4. Tornillo de regulación gruesa.
5. abrazadera.
6. Bloqueo marcha inversa.
7. Angulo marcha inversa.
8. Tornillo de regulación fina.
9. Conexión al borne de fase.
10. Conexión al borne de neutro.

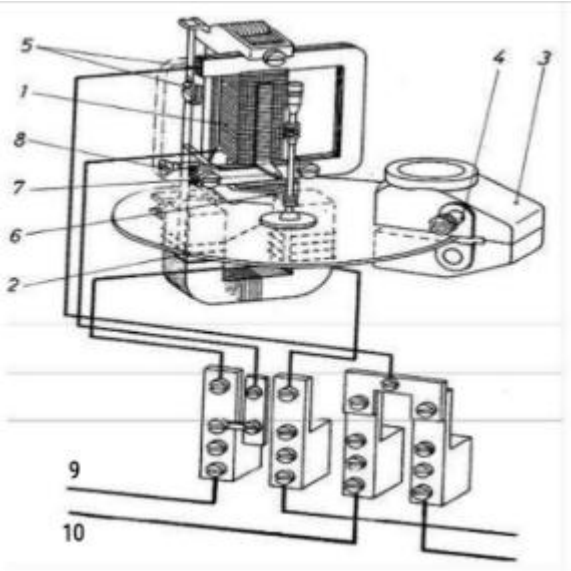


Figura 4: Partes del medidor electromecánico.

1.8.2 Medidor de energía híbrido

En la actualidad es muy poco el uso de estos instrumentos de medición, operan con una parte electrónica que generalmente es la que hace la medición y otra parte que se llama registrador el cual cuenta o acumula el consumo de energía. Esta parte es mecánica, como se muestra en la figura 5.

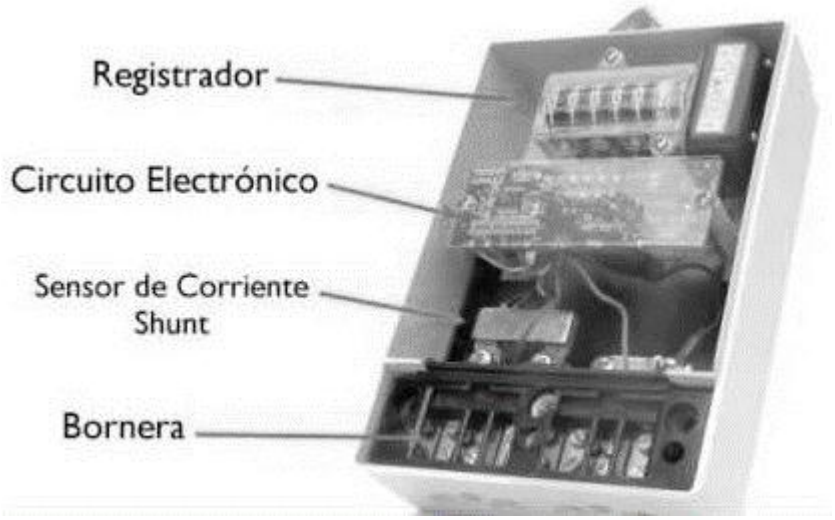


Figura 5: Medidor de energía híbrido.

1.8.3 Medidor de energía electrónico

Estos instrumentos de medición son llamados también medidores de estado sólido. Los equipos de medición han evolucionado con el tiempo al igual que muchas otras tecnologías que con los avances tecnológicos y científicos han logrado reducir el tamaño de los dispositivos o sustituirlos por otros mejorando sus prestaciones y rendimiento. En general estos instrumentos de medición están conformados por circuitos integrados microprocesadores/microcontroladores, pantallas LCD para una visualización digital, memorias para mostrar y almacenar datos. La medición de energía eléctrica con estos instrumentos se desarrolla por medio de un proceso de conversión de analógico a digital por lo que se utilizan dispositivos electrónicos para procesar la información y visualizarla, permitiendo así obtener información en formato digital, con lo cual se mejoran las mediciones y además proporcionan mucha más información útil, para poder hacer análisis tanto en el consumo de energía eléctrica, como también en los procesos de generación y distribución de energía eléctrica, con la finalidad de evaluar la calidad de la energía que llega al usuario final. En la figura 6 se muestra un medidor de energía eléctrica electrónico residencial.

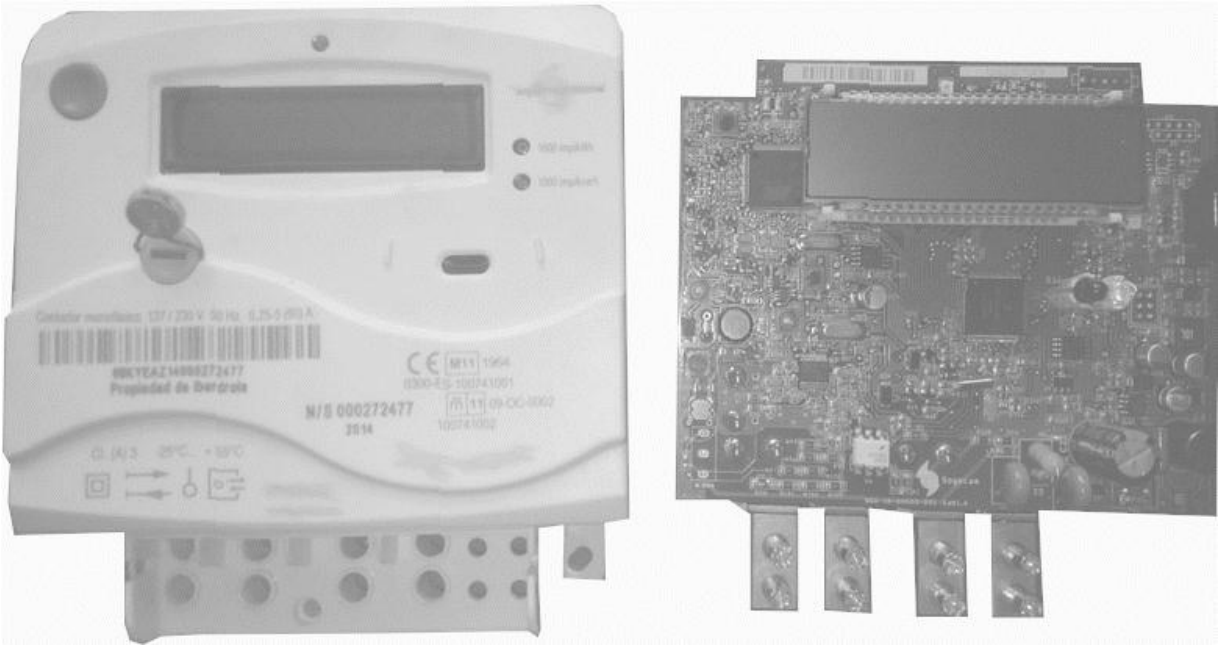


Figura 6: Medidor de energía electrónico y una vista de sus componentes internos.

1.9 Circuitos integrados medidores de energía eléctrica

En la actualidad existen varias empresas de tecnología dedicadas al desarrollo y comercialización de circuitos integrados para una amplia gama de propósitos. Este trabajo se concentra en los tipos de circuitos integrados dedicados para medir parámetros eléctricos trifásicos. Se ha seleccionado la tecnología que ofrece la empresa analog devices haciendo uso de la familia de CI ADE la cual está dividida en dos secciones, en una de ellas están las tecnologías de medición monofásica y en la otra se encuentran las tecnologías de medición polifásicas y son estas últimas las de interés para el presente trabajo en específico el CI ADE7880. La familia de CI ADE están constituidos por una combinación de convertidores analógico a digital y un DSP. Con el cual se desarrolla el procesamiento de las señales eléctricas y todos los cálculos de las mediciones de los parámetros eléctricos en cuestión.

1.9.1 Medidor de energía eléctrica polifásico ADE 7880

El circuito integrado ADE7880 de la empresa analog devices es un medidor de energía eléctrica trifásico de alta precisión, que cuenta con tres tipos de interfaces seriales para comunicación de datos y configuración además cuenta con tres salidas de pulso flexibles CF1, CF2 y CF3. El IC ADE7880 incorpora convertidores de analógico a digital (ADC) sigma-delta (Σ-Δ) de segundo orden, amplificadores PGA, integradores digitales, además incluye un circuito de referencia y todo el procesamiento de señales requerido para realizar las mediciones de energía total (fundamental y armónica) activa y aparente, cálculos rms, así como mediciones de energía activa y reactiva. Además el ADE7880 calcula el valor eficaz de los armónicos en las corrientes de fase y neutro y en los voltajes de fase, junto con las potencias activa, reactiva y aparente, el factor de potencia y la distorsión armónica de cada armónico para todas las fases, la distorsión armónica total (THD) se calcula para todas las corrientes y tensiones. Un procesador de señales digitales DSP de función fija desarrolla este procesamiento de señales. El programa que ejecuta la DSP se almacena en la memoria ROM interna. El ADE7880 es adecuado para medir energía activa, reactiva y aparente en varias configuraciones trifásicas, como servicios en estrella o delta con tres y cuatro cables. El ADE7880 proporciona funciones de calibración del sistema para cada fase, es decir, corrección de desplazamiento rms, calibración de fase y calibración de ganancia. Las salidas lógicas CF1, CF2 y CF3 proporcionan una amplia variedad de información. El ADE7880 contiene registros de muestra de forma de onda que permiten el acceso a todas las salidas de los ADC. Se pueden usar dos interfaces seriales, SPI e I2C, para comunicarse con el ADE7880. Una interfaz dedicada de alta velocidad, el puerto de captura de datos de alta velocidad (HSDC), se puede utilizar junto con I2C para proporcionar acceso a las salidas ADC e información de energía en tiempo real. El ADE7880 también tiene dos pines de solicitud de interrupción, IRQ0 e IRQ1, para indicar que ha ocurrido un evento de interrupción habilitado. Tres modos de bajo consumo especialmente diseñados garantizan la

continuidad de la acumulación de energía cuando el ADE7880 se encuentra en una situación de manipulación. El ADE7880 está disponible en el paquete LFCSP de 40 derivaciones, sin Pb como se muestra en la figura 7.

Figura 7: CI ADE7880.

Suministra información de la distorsión armónica de todos los armónicos dentro de la banda de paso de 2.8[kHz] en la corriente neutra conteniendo un error menor al 1% y para el sistema en general el error es menor al 0,1%. Con respecto al voltaje y corriente eficaces tiene un error menor al 0,1%. Algunas características relevantes son: VDD = 3,3V ± 10%, AGND = DGND = 0V, CLKIN = 16.384MHz, [Tmin – Tmax] = [−40°C a + 85°C]. Implementa la gestión de energía, la cual es controlada por medio de los pines PM0 y PM1 para el estado energético de operación del ADE7880.

Modo de suministro de energía	PM0	PM1
PSM0, modo de energía normal	1	0
PSM1, modo de potencia reducida	0	0
PSM2, modo de bajo consumo	0	1
PSM3, modo de suspensión	1	1

Tabla 1: Configuración de la gestión energética del ade7880.

Modo de energía	Todos los Registros[3]	LPOILVL, CONFIG	I2C/SPI/HSDC	Funcionalidad
PSM0 Estado después del reinicio del hardware	Establecer como predeterminado	Establecer como predeterminado	I2C habilitado	Todos los circuitos están activos y el DSP está en modo inactivo.
Estado después del reinicio del software	Establecer como predeterminado	Sin alterar	El puerto serie activo no se modifica si el procedimiento de bloqueo se ha ejecutado previamente.	Todos los circuitos están activos y el DSP está en modo inactivo.
PSM1	No disponible	Valores de PSM0 retenidos	Habilitado	Los valores absolutos medios actuales se calculan y los resultados se almacenan en los registros AIMAV, BIMAV y CIMAV. El puerto serie I2C o SPI está habilitado con funcionalidad limitada.
PSM2	No disponible	Valores de PSM0 retenidos	Deshabilitado	Compara las corrientes de fase con el umbral establecido en LPOILVL. Activa los pines IRQ0o IRQ1 en consecuencia. Los puertos serie no están disponibles.
PSM3	No disponible	Valores de PSM0 retenidos	Deshabilitado	Los circuitos internos se apagan y los puertos serie no están disponibles.

Tabla 2: Configuración interna del IC ade7880 según modo de energía.

[3] Configuración para todos los registros excepto los registros LPOILVL y CONFIG2.

Además el CI ADE7880 posee la opción de restablecerse por hardware mediante el pin 4, esta opción es válida únicamente en el modo de energía PSM0 y el pin 4 se conecta a un cero lógico 0[V] para que tenga efecto. También presenta esta capacidad de restablecerse por software mediante la manipulación del registro CONFIG el bit 7(SWRST), funciona únicamente en el modo PSM0 y se debe establecer el bit 7 aun uno lógico

1.9.1.1 Señales de entrada de voltaje y corriente

El CI ADE7880 cuenta con siete entradas analógicas las cuales se distribuyen para la adquisición de señales de corriente y señales de voltaje. Cuatro canales son destinados para las señales de corriente, que se distribuyen en las tres fases y el neutro, estos canales constan de ocho entradas o cuatro pares de entradas de voltaje diferencial. Estos pares de entradas tienen un nivel máximo de tensión de ±0.5[Vp] con respecto a AGND como se muestra en la figura 8.

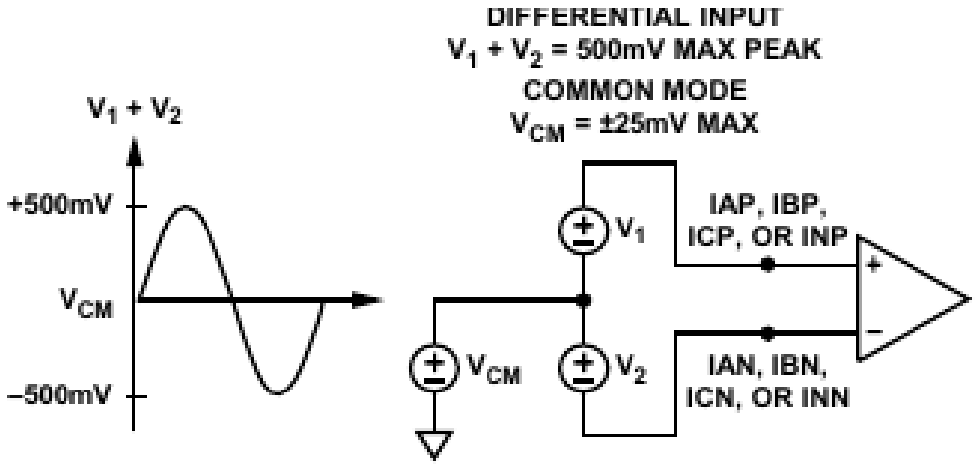


Figura 8: Nivel de tensión máximo para canales de corriente ganancia = 1.

Todas las entradas tienen un amplificador de ganancia programable (PGA) con posible selección de ganancia de 1, 2, 4, 8 y 16. La ganancia de las entradas para cada fase de corriente y la corriente del neutro se configuran en el registro GAIN en la memoria interna del DSP. Los bits [0:2] del registro configuran las ganancias para las tres corrientes de fase, los bits [3:5] configuran la ganancia para la corriente del neutro. Además cuenta con tres canales de voltaje para las tres fases y el neutro, estas entradas también tienen un nivel de tensión de entrada máximo que es de ±0.5[Vp] como se muestra en la figura 9.

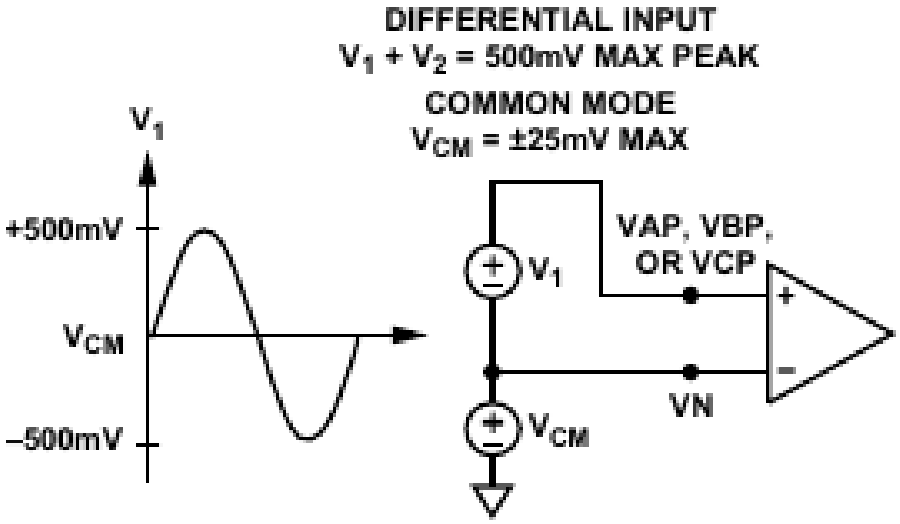


Figura 9: Nivel de tensión máximo canales de voltaje ganancia = 1.

Todas las entradas tienen un amplificador de ganancia programable con posibles selecciones de ganancias de 1, 2, 4, 8 y 16. La ganancia de las entradas para cada fase se configura en el registro GAIN, los bits [6:8] del registro configuran estas ganancias. Las señales de corrientes y voltajes de entrada se identifican como se muestra en la tabla 3.

ENTRADAS	FASE A	FASE B	FASE C	NEUTRO
CORRIENTE	IAP, IAN	IBP, IBN	ICP, ICN	INP, INN
VOLTAJE	VAP	VBP	VCP	VN

Tabla 3: Señales de entrada de voltaje y corriente por fase.

1.9.1.2 Convertidor analógico a digital (Σ-Δ)

El CI ADE7880 cuenta con siete convertidores analógico a digital sigma-delta (Σ-Δ). En el modo de operación PSM0 todos los ADC están activos. En el modo de operación PSM1 los ADC que miden las corrientes de fase están activos y los ADC que miden la corriente del neutro y los que miden los voltajes de fase se encuentran apagados. En los modos de operación PSM2 y PSM3 todos los ADC se apagan para minimizar el consumo de energía. Los ADC necesitan una referencia de voltaje. Para este diseño se utiliza la referencia de voltaje interna del CI ADE7880 que es igual a 1.2[Vdc]. En la figura 10 se muestra el diagrama en bloques de un ADC sigma-delta de primer orden. El convertidor está compuesto por el modulador Σ-Δ y el filtro digital paso bajo.

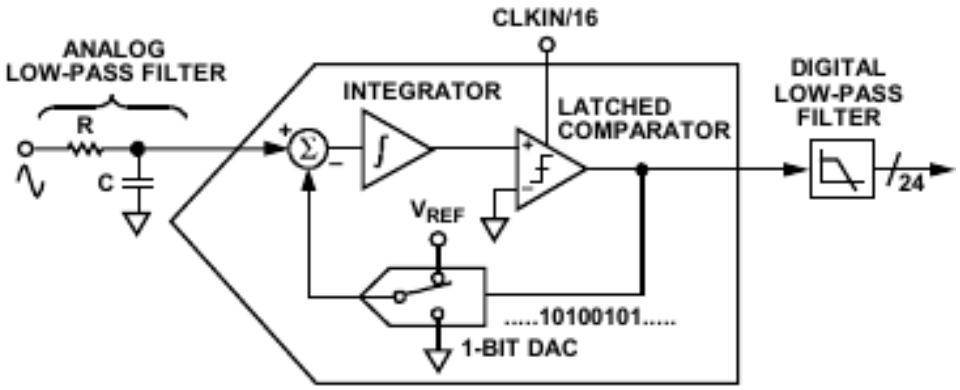


Figura 10: Convertidor de analógico a digital sigma-delta.

Un modulador Σ-Δ convierte la señal de entrada en un flujo continuo de 1 y 0 a una frecuencia determinada por el reloj de muestreo. En el ADE7880 el reloj de muestreo es igual a 1.024[MHz] ($\frac{CLKIN}{16}$). El DAC de un bit en el circuito de retroalimentación es impulsado por el flujo de datos en serie. La salida del DAC se resta de la señal de entrada. Si la ganancia del bucle es lo suficientemente alta, el valor promedio de la salida DAC (y por lo tanto, el flujo de bits) puede acercarse al del nivel de la señal de entrada. Para cualquier valor de entrada dado en un solo intervalo de muestreo, los datos del ADC de un bit son prácticamente insignificantes. Solo cuando se promedia un gran número de muestras se obtiene un resultado significativo. Este promedio se lleva a cabo en la segunda etapa del ADC, el filtro de paso bajo digital. Al promediar una gran cantidad de bits del modulador el filtro de paso bajo puede producir palabras de datos de 24 bits que son proporcionales al nivel de la señal de entrada. El convertidor Σ-Δ utiliza dos técnicas para lograr una alta resolución de lo que es esencialmente una técnica de conversión de 1 bit. El primero es el sobre muestreo. Sobre muestreo significa que la señal se muestrea a una tasa (frecuencia) que es muchas veces mayor que el ancho de banda de interés. Por ejemplo, la frecuencia de muestreo en el ADE7880 es 1.024 MHz y el ancho de banda de interés es de fMIN = 40[Hz] a fMAX = 3.3[kHz] AB = fMAX – fMIN = 3.26kHz. El sobre muestreo tiene el efecto de difundir el ruido de cuantificación (ruido debido al muestreo) en un ancho de banda más amplio. Con el ruido esparcido de manera más fina en un ancho de banda más amplio, el ruido de cuantificación en la banda de interés se reduce. Sin embargo, el sobre muestreo por sí solo no es lo suficientemente eficiente para mejorar la relación señal-ruido (SNR) en la banda de interés. Por ejemplo, se requiere un factor de sobre muestreo de 4 solo para aumentar la SNR en solo 6 dB (1 bit). Para mantener la relación de sobre muestreo a un nivel razonable, es posible dar forma al ruido de cuantificación de modo que la mayoría del ruido se encuentre en las frecuencias más altas. En el modulador Σ-Δ, el ruido es moldeado por el integrador,

que tiene una respuesta de tipo paso alto para el ruido de cuantificación. Esta es la segunda técnica utilizada para lograr una alta resolución. El resultado es que la mayor parte del ruido se encuentra en las frecuencias más altas, donde puede eliminarse mediante el filtro de paso bajo digital. Este proceso se muestra en las figuras 11 y 12.

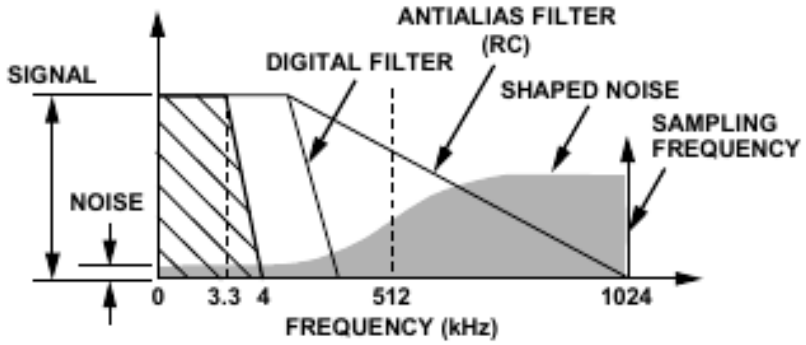


Figura 11: Modelado del ruido en modulador analógico.

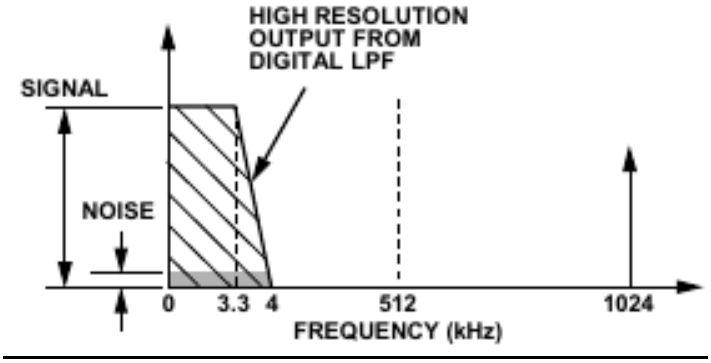


Figura 12: Reducción del ruido debido al sobre muestreo.

1.9.1.3 Procesador de señales digital (DSP)

El ADE7880 contiene un procesador de señal digital (DSP) de función fija que calcula todas las potencias y valores rms. Contiene una ROM de memoria de programa y una RAM de memoria de datos. El programa utilizado para los cálculos de potencia y rms se almacena en la memoria ROM del programa y el procesador lo ejecuta cada 8[kHz]. El final de los cálculos se indica estableciendo el Bit 17 (DREADY) en 1 en el registro STATUS0. Una interrupción adjunta a esta bandera se puede habilitar configurando el bit 17 (DREADY) en el registro MASK0. Si está habilitado, el pin IRQ0 se establece bajo y el bit de estado DREADY se establece en 1 al final de los cálculos. El bit de estado se borra y el pin IRQ0 se establece en alto escribiendo en el registro STATUS0 con el bit 17 (DREADY) establecido en 1. Los registros utilizados por el DSP se encuentran en la memoria RAM de datos, en

direcciones entre 0x4380 y 0x43BE. El ancho de esta memoria es de 28 bits. Se utiliza una canalización de dos etapas cuando se ejecutan operaciones de escritura en la memoria RAM de datos. En el momento del encendido o después de un reinicio por hardware o software, el DSP está en modo inactivo. No se ejecuta ninguna instrucción. Todos los registros ubicados en la memoria RAM de datos se inicializan en 0 o sus valores predeterminados y se pueden leer / escribir sin ninguna restricción. El registro RUN, se utiliza para iniciar y detener el DSP, se detiene al escribir 0x0000 en el registro RUN. El registro RUN debe escribirse con 0x0001 para que el DSP inicie la ejecución del código. Antes de iniciar el registro RUN debe inicializar primero todos los registros ubicados en la memoria RAM de datos con sus valores deseados y luego escriba el registro RUN con 0x0001. De esta forma, el DSP inicia los cálculos a partir de una configuración deseada. Para proteger la integridad de los datos almacenados en la memoria RAM de datos del DSP entre la dirección 0x4380 y la dirección 0x43BE, está disponible un mecanismo de protección contra escritura. Por defecto, la protección está deshabilitada y los registros ubicados entre 0x4380 y 0x43BE se pueden escribir sin restricción. Cuando la protección está habilitada, no se permiten escrituras en estos registros. Los registros se pueden leer siempre, sin restricciones, independientemente del estado de protección contra escritura. Para habilitar la protección contra escritura escriba la siguiente secuencia, 0xAD en un registro interno de 8 bits ubicado en la dirección 0xE7FE, seguido de una escritura de 0x80 en un registro interno de 8 bits ubicado en la dirección 0xE7E3. Para deshabilitar la protección contra escritura escriba la siguiente secuencia, 0xAD en un registro interno de 8 bits ubicado en la dirección 0xE7FE, seguido de una escritura de 0x00 en un registro interno de 8 bits ubicado en la dirección 0xE7E3. Se debe habilitar la protección contra escritura antes de iniciar el DSP. Si es necesario cambiar algún registro basado en RAM de memoria de datos, simplemente deshabilite la protección, cambie el valor y luego vuelva a habilitar la protección. No es necesario detener el DSP para cambiar estos registros.

1.9.1.4 Canales de entrada de corriente

La trayectoria de procesamiento de la señal para la entrada de corriente IA se muestra en la figura 13, de la misma forma se procesan las señales para las entradas de corriente IB e IC. Las salidas del ADC tienen dos signos y complementan palabras de datos de 24 bits y están disponibles a una velocidad de 8[KSPS]. Con la señal de entrada analógica a escala completa, especificada de ±0,5[Vp], el ADC produce un valor de código de salida máximo. La Figura 13 muestra una señal de voltaje a escala completa aplicada a las entradas diferenciales (IAP e IAN). La salida del ADC oscila entre −5,326,737 (0xAEB86F) y +5,326,737 (0x514791). Se debe de tener en cuenta que estos son valores nominales y cada ADE7880 varía alrededor de estos valores.

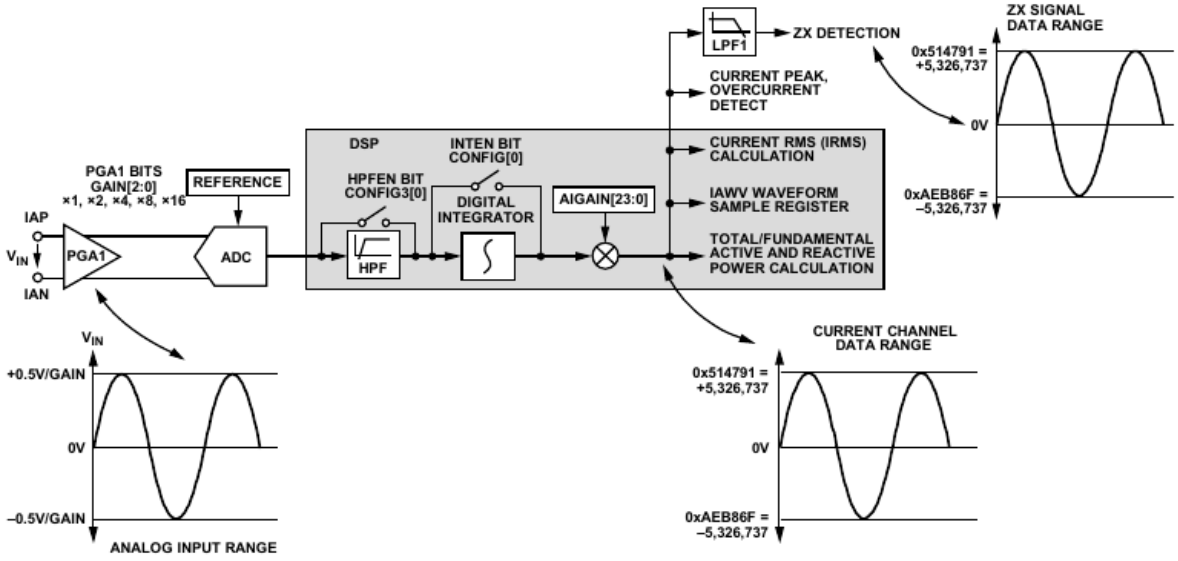


Figura 13: Trayectoria del procesamiento la señal para los canales de corriente.

La entrada, IN, corresponde a la corriente neutra de un sistema trifásico. Si no hay una línea neutra, conecte esta entrada a AGND. La trayectoria de procesamiento de la señal para la corriente neutra es similar a la trayectoria de las corrientes de fase como se muestra en la Figura 14.

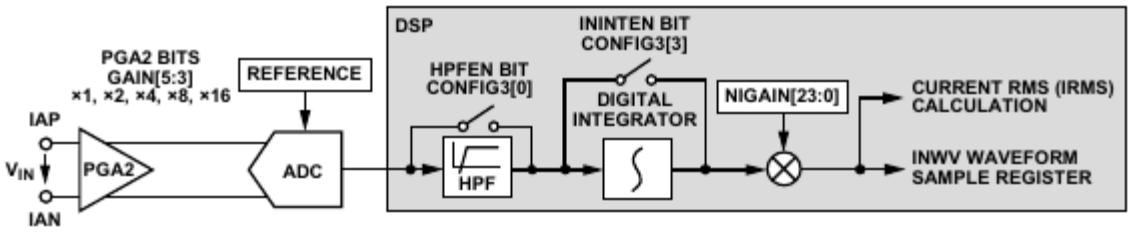


Figura 14: Trayectoria del procedimiento la señal para el canal de corriente neutra.

1.9.1.4.1 Registros de ganancia de forma de onda de corriente

En la trayectoria de la señal de cada fase de corriente se encuentra un multiplicador, de la misma forma para la señal de corriente neutra. La forma de onda de corriente se puede cambiar en ± 100% escribiendo un número de complemento a dos correspondiente en los registros de ganancia de forma de onda de corriente de 24 bits con signo AIGAIN, BIGAIN, CIGAIN y NIGAIN. Por ejemplo, si se escribe 0x400000 en esos registros, la salida de ADC aumenta en un 50%. Para escalar la entrada en −50%, escriba 0xC00000 en los registros.

Cambiar el contenido de los registros AIGAIN, BIGAIN, CIGAIN o INGAIN afecta todos los cálculos basados en su corriente; es decir, afecta a la energía activa, reactiva y aparente de cada fase y el cálculo rms de corriente. Además, las muestras de formas de onda se escalan en consecuencia. Tenga en cuenta que los puertos serie del ADE7880 funcionan con palabras de 32, 16 u 8 bits, y el DSP funciona con 28 bits. Se accede a los registros AIGAIN, BIGAIN, CIGAIN y NIGAIN de 24 bits como registros de 32 bits con los cuatro bits más significativos (MSB) rellenados con 0 y el signo extendido a 28 bits. Como se muestra en la figura 15.

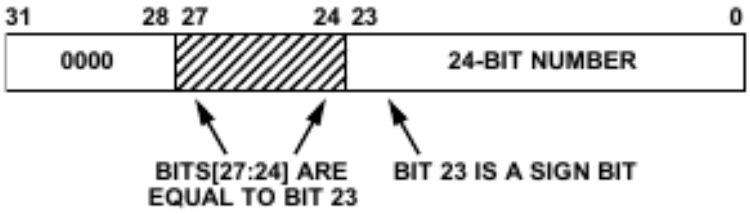


Figura 15: Registros xIGAIN transmitido como palabras de 32 bits.

1.9.1.4.2 Filtro paso alto digital (HPF) del canal de corriente

Las salidas del ADC pueden contener un desplazamiento de CC. Esta compensación puede generar errores en los cálculos de potencia y rms. Los filtros de paso alto digital (HPF) se colocan en la trayectoria de las señales de las corrientes de fase y neutra y de los voltajes de fase. Si está habilitado, el HPF elimina cualquier desplazamiento de CC en el canal de corriente. Todos los filtros están implementados en el DSP y, por defecto, todos están habilitados: El bit 0 (HPFEN) del registro CONFIG3 [7: 0] se establece en 1. Todos los filtros se inhabilitan configurando el bit 0 (HPFEN) en 0.

1.9.1.4.3 Muestreo del canal de corriente

Las muestras de forma de onda del canal de corriente se toman en la salida del HPF y se almacenan en los registros firmados de 24 bits, IAWV, IBWV, ICWV e INWV a una velocidad de 8 [kSPS]. Todos los cálculos de potencia y rms permanecen ininterrumpidos durante este proceso. El bit 17 (DREADY) en el registro STATUS0 se establece cuando los registros IAWV, IBWV, ICWV e INWV están disponibles para leerse utilizando el puerto serie I2C o SPI. Los puertos seriales del ADE7880 funcionan en palabras de 32, 16 u 8 bits. Cuando los registros firmados IAWV, IBWV, ICWV e INWV de 24 bits se leen del ADE7880, se transmiten con el signo extendido a 32 bits. Como se muestra en la Figura 16.

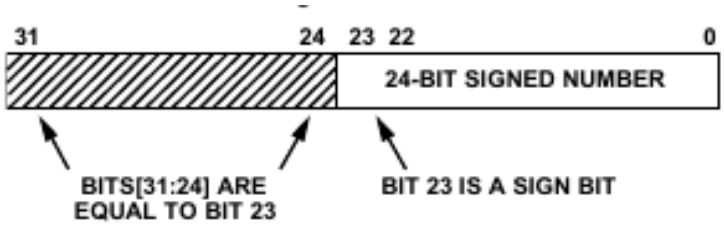


Figura 16: Registro IxWV transmitido como palabra firmada de 32 bits.

1.9.1.5 Canales de entrada de voltaje

La Figura 17 muestra la trayectoria de procesamiento para la señal de entrada del canal de voltaje VA. Los canales de voltaje VB y VC tienen trayectorias de procesamiento de señal similares. Las salidas del ADC tienen dos signos y complementan palabras de datos de 24 bits que están disponibles a una velocidad de 8[KSPS]. Con la señal de entrada analógica a escala completa especificada de ± 0,5[Vp], el ADC produce un valor de código de salida máximo. La Figura 17 muestra una señal de voltaje a escala completa que se aplica a las entradas diferenciales (VA y VN). La salida de ADC oscila entre -5,326,737 (0xAEB86F) y +5,326,737 (0x514791). Se debe tener en cuenta que estos son valores nominales y cada ADE7880 varía alrededor de estos valores.

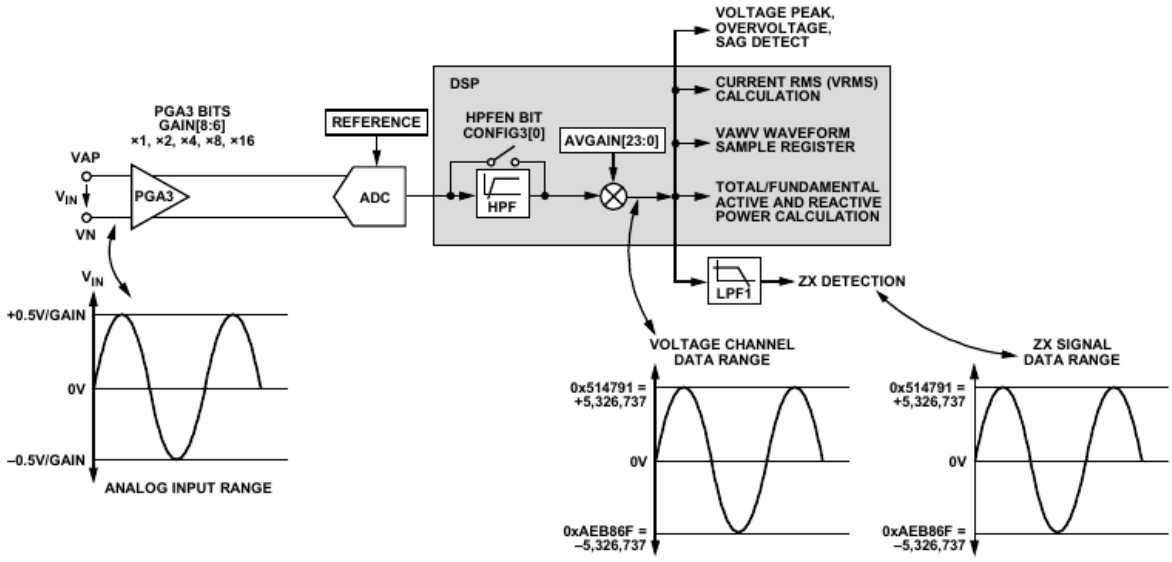


Figura 17: Trayectoria del procesamiento de la señal para los canales de voltaje.

1.9.1.5.1 Registros de ganancia de forma de onda de voltaje

En la trayectoria de la señal de cada fase de voltaje se encuentra un multiplicador. La forma de onda de voltaje se puede cambiar en ± 100% escribiendo un número de complemento a dos correspondiente en los registros de ganancia de forma de onda de voltaje con signo de 24 bits AVGAIN, BVGAIN y CVGAIN. Por ejemplo, si se escribe 0x400000 en esos registros, la salida de ADC aumenta en un 50%. Para escalar la entrada en −50%, escriba 0xC00000 en los registros. Cambiar el contenido de los registros AVGAIN, BVGAIN y CVGAIN afecta todos los cálculos basados en sus voltajes; es decir, afecta a las potencias de fase activa, reactiva y aparente así como también el cálculo del voltaje rms. Además, las muestras de forma de onda se escalan en consecuencia. Los puertos seriales del ADE7880 funcionan con palabras de 32, 16 u 8 bits, y el DSP funciona con 28 bits. Como se muestra en la figura 15, se accede a los registros AVGAIN, BVGAIN y CVGAIN como registros de 32 bits con los cuatro MSB rellenados con ceros y el signo extendido a 28 bits.

1.9.1.5.2 Filtro paso alto digital (HPF) del canal de voltaje

Como se explica en la sección 1.9.1.4.2, las salidas del ADC pueden contener una compensación de CC que puede generar errores en los cálculos de potencia y rms. Los HPF se colocan en la ruta de la señal de los voltajes de fase, similares a los de los canales de corriente. El bit 0 (HPFEN) del registro CONFIG3 puede habilitar o deshabilitar los filtros. Consulte la sección 1.9.1.4.2 para obtener más detalles.

1.9.1.5.3 Muestreo del canal de voltaje

Las muestras de forma de onda del canal de voltaje se toman en la salida del HPF y se almacenan en los registros VAWV, VBWV y VCWV de 24 bits con signo a una velocidad de 8[KSPS]. Todos los cálculos de potencia y rms permanecen ininterrumpidos durante este proceso. El bit 17 (DREADY) en el registro STATUS0 se establece cuando los registros VAWV, VBWV y VCWV están disponibles para leerse mediante el puerto serie I2C o SPI. Como se indica en la sección 1.9.1.4.1, los puertos seriales del ADE7880 funcionan con palabras de 32, 16 u 8 bits. De manera similar a los registros presentados en la figura 16, los registros con signo VAWV, VBWV y VCWV de 24 bits se transmiten con signo extendido a 32 bits.

1.9.1.6 Medición del período de la fase de voltaje

El ADE7880 proporciona medición del período de la línea del canal de voltaje. El período de cada voltaje de fase se mide y se almacena en tres registros diferentes, APERIOD, BPERIOD y CPERIOD. Los registros del período son registros sin firmar de 16 bits y se

actualizan cada período de línea. Debido al filtro LPF1 (consulte la figura 17), se asocia un tiempo de estabilización de 30[ms] a 40[ms] con este filtro antes de que la medición sea estable. La medición del período tiene una resolución de 3.90625[µs/LSB] (reloj de 256 kHz), que representa 0.0195% (50 Hz / 256 kHz) cuando la frecuencia de línea es 50 Hz y 0.0234% (60 Hz / 256 kHz) cuando la frecuencia de línea es 60 Hz. El valor de los registros del período para redes de 50 Hz es aproximadamente 5120 (256 kHz / 50 Hz) y para redes de 60 Hz es aproximadamente 4267 (256 kHz / 60 Hz). La longitud de los registros permite la medición de frecuencias de línea tan bajas como 3,9 Hz (256 kHz / 216). Los registros del período son estables a ± 1 LSB cuando se establece la línea y la medición no cambia. Las siguientes ecuaciones se pueden utilizar para calcular el período y la frecuencia de la línea utilizando los registros de período: $\boldsymbol{T_L = \frac{xPERIOD[15:0]}{256E3}[seg]}$ $\boldsymbol{Ecuación\ 1.8}$

$$\boldsymbol{f_L = \frac{256E3}{xPERIOD[15:0]}[Hz]} \quad \boldsymbol{Ecuación\ 1.9}$$

1.9.1.7 Registros de compensación de fase

Como se describe en las secciones 1.9.1.4 y 1.9.1.5, la ruta de La trayectoria de procesamiento de la señal de corriente y el voltaje son similares. El error de fase entre las señales de corriente y tensión introducidas por el ADE7880 es insignificante. Sin embargo, el ADE7880 debe funcionar con transductores que pueden introducir errores de fase inherentes. Por ejemplo, un transformador de corriente (TC) con un error de fase de 0,1° a 3° no es infrecuente. Estos errores de fase pueden variar de una pieza a otra y deben corregirse para realizar cálculos de potencia precisos. Los errores asociados con el desajuste de fase son particularmente notables en factores de potencia bajos. El ADE7880 proporciona un medio para calibrar digitalmente estos pequeños errores de fase. El ADE7880 permite que se introduzca un pequeño retardo o avance de tiempo en la trayectoria de procesamiento de señales para compensar los pequeños errores de fase. Los registros de calibración de fase APHCAL, BPHCAL y CPHCAL son registros de 10 bits que pueden variar el avance de tiempo en la trayectoria de la señal del canal de voltaje de -374.0[µs] a +61.5[µs]. Los valores negativos escritos en los registros PHCAL representan un avance de tiempo, mientras que los valores positivos representan un retraso de tiempo. Un LSB equivale a 0,976[µs] de retardo o avance de tiempo con una frecuencia de reloj de 1.024 MHz. Con una frecuencia de línea de 60 Hz, da una resolución de fase de 0.0211°(360° × 60Hz / 1.024MHz) en la fundamental. Esto corresponde a un rango de corrección total de −8.079° a +1.329° a 60 Hz. A 50 Hz, el rango de corrección es de −6.732° a +1.107° y la resolución es de 0.0176°(360° × 50Hz / 1,024MHz). Dado un error de fase de x grados, medido usando el voltaje de fase como referencia, los LSB correspondientes se calculan dividiendo x por la resolución de fase (0.0211° / LSB para 60 Hz y 0.0176° / LSB para 50 Hz). Los resultados entre −383 y +63 son aceptables sin embargo no se aceptan números fuera de este rango. Si la corriente

adelanta al voltaje, el resultado es negativo y el valor absoluto se escribe en los registros PHCAL. Si la corriente se retrasa respecto al voltaje, el resultado es positivo y se agrega 512 al resultado antes de escribirlo en xPHCAL.

$$APHCAL, BPHCAL, CPHCAL = \left\{\left|\frac{x}{resolucion\ de\ fase}\right|\right\}, x \leq 0 \quad Ecuación\ 1.10$$

$$\left\{\frac{x}{resolucion\ de\ fase} + 512\right\}, x > 0 \quad Ecuación\ 1.11$$

La figura 19 ilustra cómo se usa la compensación de fase para eliminar x = -1° en el canal de corriente de fase A debido transformador de corriente (equivalente a 55,5 µs para sistemas de 50 Hz). Los puertos seriales del ADE7880 funcionan en palabras de 32, 16 u 8 bits. Como se muestra en la figura 18, se accede a los registros APHCAL, BPHCAL y CPHCAL de 10 bits como registros de 16 bits con los seis bits MSB rellenados con ceros.

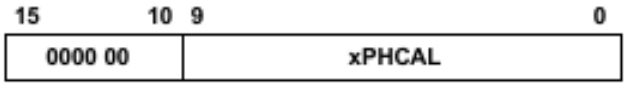


Figura 18: Registros xPHCAL transmitidos como registros de 16 bits.

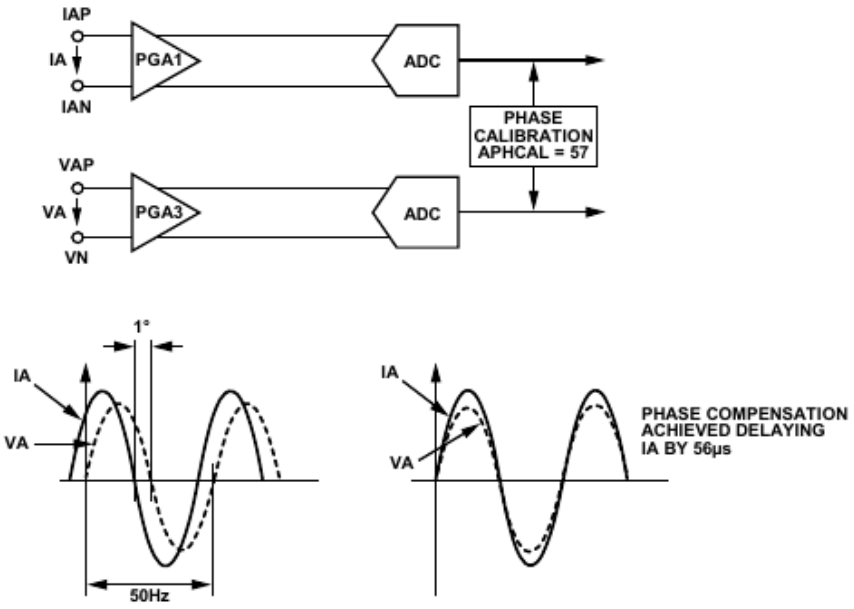


Figura 19: Calibración de fase en canales de corriente.

1.9.1.8 Circuito de referencia interna de voltaje

El voltaje de referencia nominal en el pin (17) REFIN / OUT es de 1.2[V]. Este es el voltaje de referencia usado para los ADC en el ADE7880. Si se desea utilizar un voltaje de referencia externo típico de 1.2[V] para saturar el pin REFIN / OUT. Calcule el coeficiente de temperatura de la referencia de voltaje interno según el método de punto final. Para calcular la derivada sobre la temperatura, mida y compare los valores de la referencia de voltaje en los puntos finales (−40°C y +85°C) con el valor de referencia a 25°C para obtener la pendiente de la curva del coeficiente de temperatura. La figura 20 es una representación típica de la derivada sobre la temperatura.

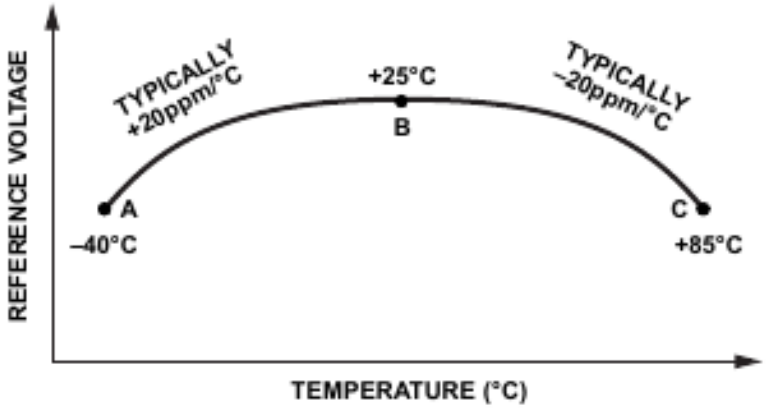


Figura 20: Desviación de temperatura para el circuito de referencia interna.

Debido a que la referencia se usa para todos los ADC, cualquier desviación del x% en la referencia da como resultado una desviación del 2x% de la precisión del medidor. La derivada de referencia resultante de los cambios de temperatura suele ser muy pequeña y por lo general, mucho menor que la derivada de otros componentes en un medidor. Alternativamente, calibre el medidor a múltiples temperaturas. El ADE7880 usa la referencia de voltaje interna cuando el Bit 0 (EXTREFEN) en el registro CONFIG2 se borra a 0 el valor predeterminado; la referencia de voltaje externo se usa cuando el bit está establecido en 1. Establezca el registro CONFIG2 durante el modo PSM0; su valor se mantiene durante los modos de energía PSM1, PSM2 y PSM3.

1.9.1.9 Medición del valor eficaz (RMS)

La raíz cuadrada media o RMS es una medida de la magnitud de una señal de CA. Su definición puede ser tanto práctica como matemática. Definiendo en la práctica el valor rms asignado a una señal de CA es la cantidad de CC requerida para producir una cantidad

equivalente de potencia en la carga. Matemáticamente, el valor rms de una señal continúa f(t) se define como: $\boldsymbol{F_{RMS}} = \sqrt{\frac{1}{t} * \int_0^t \boldsymbol{f^2}\,(\boldsymbol{t})\boldsymbol{dt}}$ $\boldsymbol{Ecuación\ 1.12}$ Esta ecuación implica que para las señales que contienen armónicos, el cálculo rms contiene la contribución de todos los armónicos, no solo el fundamental. El ADE7880 utiliza dos métodos diferentes para calcular los valores rms. El primero es muy preciso y está activo solo en modo PSM0. El segundo es menos preciso, utiliza la estimación de la medición del valor absoluto medio (mav) y está activo en los modos PSM0 y PSM1. El ADE7880 también calcula los valores rms de varios componentes fundamentales y armónicos de corrientes de fase, voltajes de fase y corriente neutra como parte del bloque de cálculo de armónicos. El primer método consiste en filtrar el cuadrado de la señal de entrada con un filtro digital paso bajo (LPF) y al resultado se le extrae la raíz cuadrada para mayor de talle consulte la figura 21. El cálculo rms basado en este método se procesa simultáneamente en los siete canales de entrada analógica. Cada resultado está disponible en los registros de 24 bits: AIRMS, BIRMS, CIRMS, AVRMS, BVRMS, CVRMS y NIRMS.

1.9.1.10 Cálculo de corriente rms

La figura 21 muestra el detalle de la trayectoria de procesamiento de la señal para el cálculo rms en una de las fases de los canales de corriente. El valor rms del canal de corriente se procesa a partir de las muestras utilizadas en el canal de corriente. Los valores de corriente rms son valores de 24 bits con signo y se almacenan en los registros AIRMS, BIRMS, CIRMS, NIRMS. La tasa de actualización de la medición de corriente rms es de 8[kHz]. Si el bit 2 (INSEL) del registro CONFIG3 es 0 (predeterminado), el registro NIRMS contiene el valor rms de la corriente neutra. Si el bit INSEL es 1, el registro NIRMS contiene el valor eficaz de la suma de los valores instantáneos de las corrientes de fase. Con la señal de entrada analógica de escala completa especificada de ±0,5[Vp], el ADC produce un código de salida que es aproximadamente ± 5,326,737. El valor eficaz equivalente de una señal sinusoidal a escala completa es 3,766,572 (0x39792C), independientemente de la frecuencia de la línea. La precisión de la corriente rms es típicamente un error de 0.1% desde la entrada de escala completa hasta 1/1000 de la entrada de escala completa cuando PGA = 1. Además, esta medición tiene un ancho de banda de 3.3kHz. La Tabla 4 muestra el tiempo de establecimiento para la medición de la corriente rms, que es el tiempo que tarda el registro xIRMS en establecerse dentro del 99,5% de la entrada al canal de corriente cuando se inicia desde 0 hasta la escala completa. Sin embargo, durante los casos de encendido del CI y reinicio del DSP, normalmente se necesitan alrededor de 1,3 segundos desde el momento en que el registro RUN se establece en 0x01 para que una señal FS / 1000 se establezca en un 99,5%.

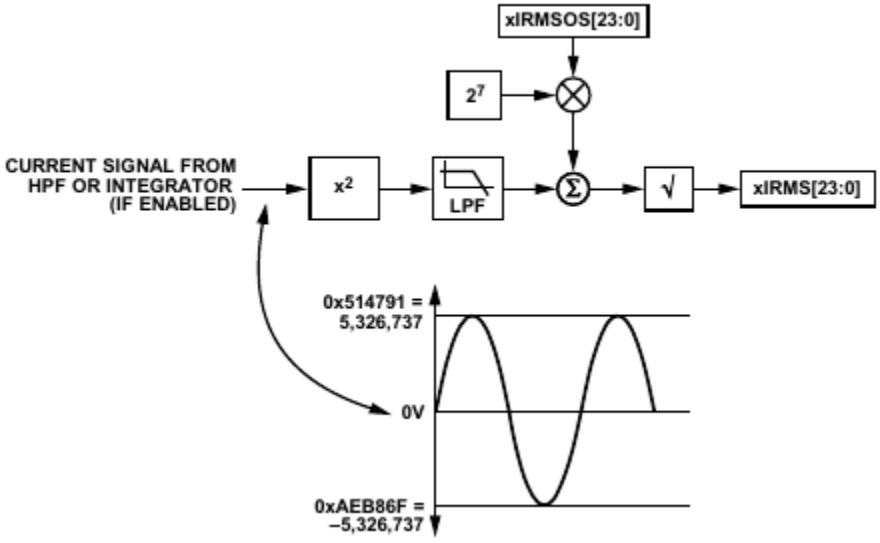


Figura 21: Procesamiento de las señales de corriente rms.

Estado del integrador	Señal de entrada a 50Hz	Señal de entrada a 60Hz
Integrador apagado	580[ms]	580[ms]
Integrador encendido	700[ms]	700[ms]

Tabla 4: Tiempo de ajuste para la medición de corriente rms.

1.9.1.10.1 Compensación de corriente rms

El ADE7880 incorpora un registro de compensación de desplazamiento para la corriente rms en cada fase: AIRMSOS, BIRMSOS, CIRMSOS y NIRMSOS. Estos son registros con signo de 24 bits que se utilizan para eliminar compensaciones en los cálculos de corriente rms. Puede existir una compensación en el cálculo rms debido a los ruidos de entrada que están integrados en el componente de cd del i2(t). Realice la calibración de compensación a baja corriente; Evite el uso de corrientes iguales a cero para este propósito. De manera similar al registro presentado en la figura 15, se accede a los registros firmados de 24 bits AIRMSOS, BIRMSOS, CIRMSOS y NIRMSOS como registros de 32 bits con los cuatro bits MSB rellenados con 0 y el signo extendido a 28 bits.

1.9.1.11 Cálculo de voltaje rms

La figura 22 muestra el detalle de la trayectoria de procesamiento de señales para el cálculo rms en una de las fases del canal de tensión. El valor rms del canal de voltaje se procesa a partir de las muestras utilizadas en el canal de voltaje. Los valores rms de voltaje son valores de 24 bits con signo y se almacenan en los registros AVRMS, BVRMS y CVRMS. La tasa de actualización de la medición rms es de 8[kHz]. Con la señal de entrada analógica de escala completa especificada de ±0,5[Vp], el ADC produce un código de salida que es aproximadamente ±5,326,737. El valor eficaz equivalente de una señal sinusoidal a escala completa es 3,766,572 (0x39792C), independientemente de la frecuencia de la línea. La precisión del voltaje rms es típicamente de 0.1% de error desde la entrada de escala completa hasta 1/1000 de la entrada de escala completa. Adicionalmente,

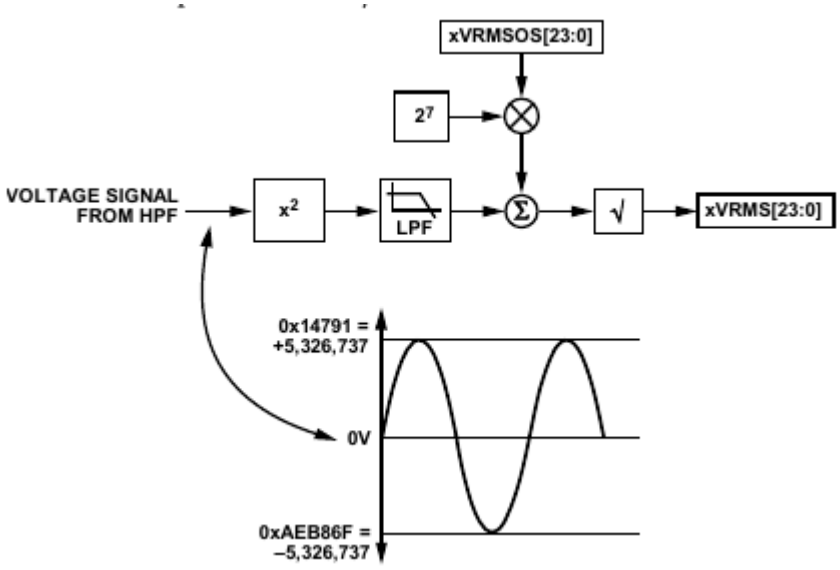


Figura 22: Procesamiento de las señales de voltaje rms.

Esta medida tiene un ancho de banda de 3,3 kHz. El tiempo de establecimiento de la medición V[rms] es de 580 ms para las señales de entrada de 50 Hz y 60 Hz. El tiempo de estabilización de la medición V[rms] es el tiempo que tarda el registro en reflejar el valor de la entrada del canal de voltaje cuando se inicia desde 0. Se accede a los registros AVRMS, BVRMS y CVRMS firmados de 24 bits como registros de 32 bits con los ocho bits MSB rellenados con ceros.

1.9.1.11.1 Compensación de voltaje rms

El ADE7880 incorpora registros de compensación de voltaje rms para cada fase: AVRMSOS, BVRMSOS y CVRMSOS. Estos son registros con signo de 24 bits que se utilizan para eliminar compensaciones en los cálculos de voltaje rms. Puede existir una compensación en el cálculo rms debido a ruidos de entrada que están integrados en el componente de cd de V2(t). Realice la calibración de compensación a baja corriente; Evite el uso de voltajes iguales a cero para este propósito. Se accede a los registros firmados de 24 bits AVRMSOS, BVRMSOS, CVRMSOS como registros de 32 bits con los cuatro bits MSB rellenados con 0 y el signo extendido a 28 bits.

1.9.1.12 Cálculo de la potencia activa

El ADE7880 calcula la potencia activa total en cada fase. La potencia activa total considera en su cálculo los componentes fundamentales y armónicos de las tensiones y corrientes. Además, el ADE7880 calcula la potencia activa fundamental, esta potencia es determinada solo por los componentes fundamentales de los voltajes y corrientes. El ADE7880 también calcula las potencias activas armónicas, las potencias activas determinadas por las componentes armónicas de los voltajes y corrientes. La energía eléctrica se define como la tasa de flujo de energía desde la fuente a hacia la carga, está dada por el producto de las formas de onda de voltaje y corriente. La forma de onda resultante se denomina señal de potencia instantánea y es igual a la tasa de flujo de energía en cada instante de tiempo. Si un sistema de CA recibe un voltaje, v(t), y consume una corriente, i(t), y cada uno de ellos contiene armónicos, entonces se pueden definir las siguientes expresiones matemáticas respectivamente.

$$\boldsymbol{P = \frac{1}{nT} * \int_{0}^{nT} p(t)dt \; = \sum_{k=1}^{\infty} V_k * I_k * cos(\varphi - \gamma) \quad Ecuación\ 1.13}$$

Dónde:

- T = es el período del ciclo de línea.
- P = es la potencia activa total.

Esta es la ecuación utilizada para calcular la potencia activa total en el ADE7880 para cada fase. La ecuación de potencia activa fundamental se obtiene de la Ecuación 1.13 con k = 1, que evaluándola nos queda como sigue:

$$\boldsymbol{P_{fundamental} = V_1 * I_1 * cos(\varphi_1 - \gamma_1) \quad Ecuación\ 1.14}$$

La figura 23 muestra cómo el ADE7880 calcula la potencia activa total en cada fase. Primero, multiplica las señales de corriente y voltaje en cada fase. A continuación, extrae el componente de CC de la señal de potencia instantánea en cada fase PA, PB y PC utilizando el LPF filtro de paso bajo digital. Si las corrientes y tensiones de fase contienen solo el componente fundamental, están en fase es decir, $\varphi 1 = \gamma 1 = 0$ y corresponden a entradas del ADC a escala completa, multiplicarlas da como resultado una señal de potencia instantánea que tiene un componente de cd, $V1 \times I1$, y un componente sinusoidal, $V1 \times I1 \cos(2\omega t)$; La figura 25 muestra las formas de onda correspondientes.

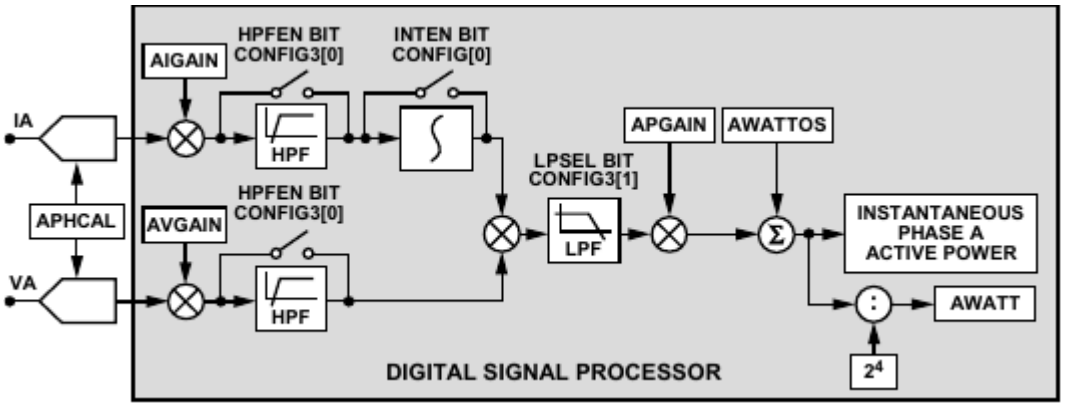


Figura 23: Trayectoria de procesamiento de la señal para la potencia activa.

Debido a que LPF no tiene una respuesta de frecuencia ideal, la señal de potencia activa tiene cierta ondulación en a la señal de potencia instantánea. Esta ondulación es sinusoidal y tiene una frecuencia igual al doble de la frecuencia de la línea. Dado que la ondulación es de naturaleza sinusoidal, se elimina cuando la señal de potencia activa se integra a lo largo del tiempo para calcular la energía. El bit 1 (LPFSEL) del registro CONFIG3 selecciona el tiempo de estabilización y la atenuación del LPF. Si LPFSEL es 0 predeterminado, el tiempo de estabilización es 650 ms y la atenuación de la ondulación es 65 dB. Si LPFSEL es 1, el tiempo de estabilización es 1300 ms y la atenuación de la ondulación es 128 dB. El ADE7880 almacena las potencias activas totales de fase en los registros AWATT, BWATT y CWATT de 24 bits.

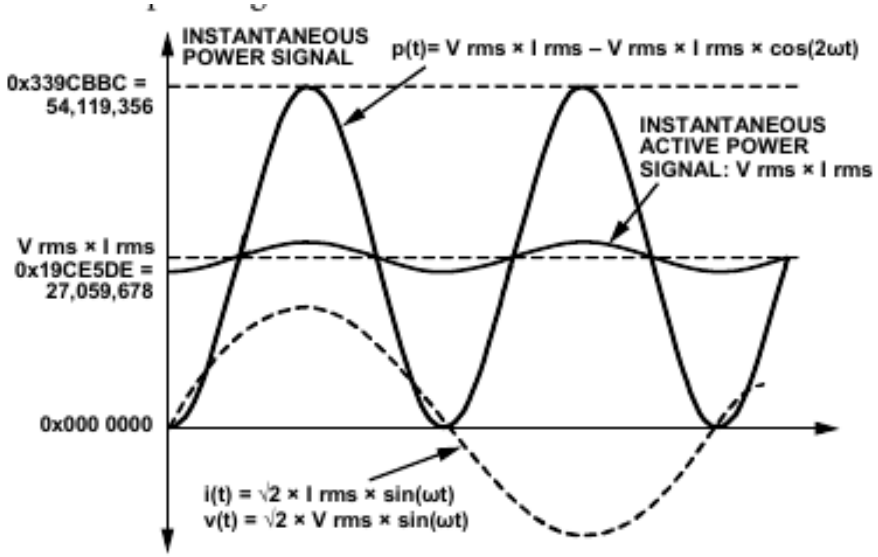


Figura 24: Cálculo de potencia activa a plena escala.

1.9.1.12.1 Registro para el cálculo de potencias fundamentales

El ADE7880 calcula la potencia activa fundamental utilizando un algoritmo patentado que requiere algunas inicializaciones en función de la frecuencia de la red y su voltaje nominal medido en el canal de voltaje. El bit 14 (SELFREQ) en el registro COMPMODE debe configurarse de acuerdo con la frecuencia de la red en la que está conectado el ADE7880. Si la frecuencia de la red está entre 45 Hz y 55 Hz, borre este bit a 0 (el valor predeterminado). Si la frecuencia de la red está entre 55 Hz y 66 Hz, establezca este bit en 1. Además, inicialice el registro con signo VLEVEL de 28 bits según la siguiente ecuación:

$$\boldsymbol{V_{LEVEL} = \frac{V_{FS}}{V_n} * 4x10^6 \quad Ecuación\ 1.15}$$

Dónde:

VFS = al valor rms del voltaje de fase cuando las entradas al ADC están a escala completa.

Vn = al valor nominal eficaz de la tensión de fase.

El registro VLEVEL es un registro con signo de 28 bits y se rellena con ceros a 32 bits.

1.9.1.12.2 Calibración de ganancia de potencia activa

Tenga en cuenta que el resultado de la potencia activa promedio a la salida LPF en cada fase se puede escalar en ± 100% escribiendo en los registros de 24 bits de ganancia para cada fase APGAIN, BPGAIN, CPGAIN. Los registros xPGAIN se colocan en las trayectorias de procesamiento de señal de todas las potencias calculadas por el ADE7880: potencias activas totales, potencias activas y reactivas fundamentales y potencias aparentes. Esto es posible porque todas las trayectorias de procesamiento de señales de energía tienen ganancias generales idénticas. Por lo tanto, para compensar los errores de ganancia en las potencias, es suficiente analizar solo una trayectoria de datos de potencia. Los registros de ganancia para las potencias son registros en complemento a dos, con signo. La salida se escala en -50% escribiendo 0xC00000 en los registros de ganancia, y se aumenta en + 50% escribiendo 0x400000 en ellos. Estos registros se utilizan para calibrar el cálculo de la potencia o energía activa, reactiva y aparente para cada fase. De manera similar a los registros presentados en la figura 15, se accede a los registros xPGAIN.

1.9.1.12.3 Calibración de compensación de la potencia activa

El ADE7880 incorpora un registro de 24 bits de compensación en cada fase para la potencia activa. Los registros AWATTOS, BWATTOS y CWATTOS son utilizados en la compensación de los cálculos de potencia activa total, y los registros AFWATTOS, BFWATTOS y CFWATTOS para compensar los cálculos de potencia activa fundamental. Estos son registros de 24 bits en complemento a dos con signo. Puede existir un desplazamiento en el cálculo de potencia debido a la diafonía entre canales en la PCB o en el propio chip. De manera similar a los registros presentados en la figura 15, se accede a los registros con signo de 24 bits xWATTOS y xFWATTOS.

1.9.1.12.4 Cálculo de energía activa

Como se indicó anteriormente, la potencia se define como la tasa de flujo de energía. Esta relación se puede expresar matemáticamente como: $P = \frac{dEnergia}{dt}$ Por el contrario, la energía se da como la integral de la potencia, como sigue: $Enegia_{Activa} = \int p(t)dt$ El ADE7880 hace la integración de la señal de potencia activa en dos etapas según se muestra en la figura 25. El proceso es idéntico para las potencias activas totales y fundamentales. La primera etapa acumula la potencia activa total o fundamental de fase a 1.024MHz, aunque son calculadas por el DSP a una tasa de 8[kHz]. Cada vez que se alcanza un umbral, se genera un pulso y el umbral se resta del registro interno. La segunda etapa consiste en acumular los pulsos generados en la primera etapa en registros internos de acumulación de 32 bits. El contenido de estos registros se transfiere a los registros, xWATTHR y xFWATTHR, cuando se accede a estos registros.

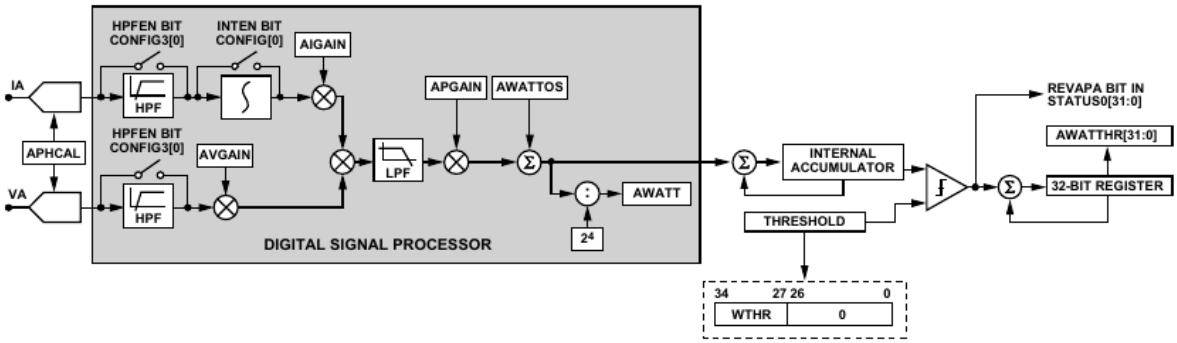


Figura 25: Trayectoria de acumulación de energía activa total.

La figura 26 explica este proceso. El umbral se forma concatenando el registro sin signo de 8 bits WTHR a 27 bits iguales a 0. Este valor es introducido por el usuario y es común para las potencias activas totales y fundamentales en todas las fases. Su valor depende de cuánta energía se asigne a un LSB de los registros de vatios-hora. WTHR.

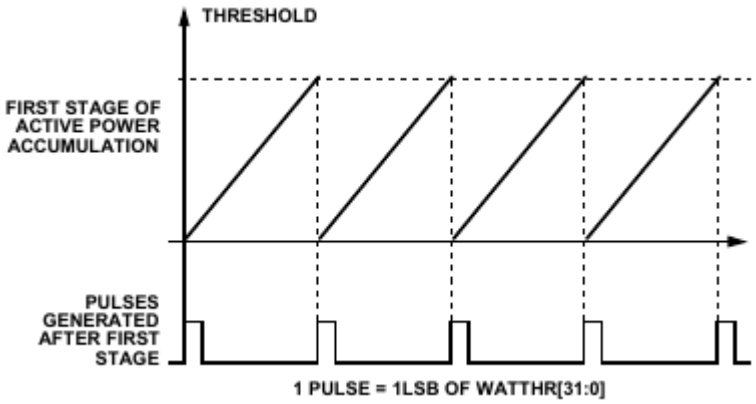


Figura 26: Acumulación de potencia activa dentro del DSP.

El registro WTHR es un número sin signo de 8 bits, por lo que su valor máximo es 2^8 - 1. Su valor predeterminado es 0x3. Evite valores inferiores a 3, y bajo ninguna circunstancia utilice 0, ya que el umbral. Debe ser un valor distinto de cero. Esta acumulación o suma de tiempo discreto es equivalente a la integración en tiempo continuo siguiendo la descripción de la Ecuación 1.16: $\boldsymbol{ENERGIA = \int P(t)dt = \lim_{T\to 0} \sum_{n=0}^{\infty} P(nT) * T}$ **Ecuación 1.16** dónde:

n = al número de muestra de tiempo discreto.

T = al período.

En el ADE7880, las energías activas de fase totales se acumulan en los registros firmados AWATTHR, BWATTHR y CWATTHR de 32 bits, y las energías activas de fase fundamental se acumulan en los registros firmados AFWATTHR, BFWATTHR y CFWATTHR de 32 bits. El contenido del registro de energía activa puede pasar a negativo a escala completa (0x80000000) y continuar aumentando de valor cuando la potencia activa es positiva. Por el contrario, si la potencia activa es negativa, el registro de energía se desborda a la escala completa positiva (0x7FFFFFFF) y continúa disminuyendo de valor. El ADE7880 proporciona una bandera de estado para señalar cuando uno de los registros xWATTHR y xFWATTHR está medio lleno. El bit 0 (AEHF) en el registro STATUS0 se establece cuando el bit 30 de uno de los registros xWATTHR cambia, lo que significa que uno de estos registros está medio lleno. Si la potencia activa es positiva, el registro de vatios-hora se llena a la mitad cuando aumenta de 0x3FFF FFFF a 0x4000 0000. Si la potencia activa es negativa, el registro de vatios-hora se llena a la mitad cuando disminuye de 0xC000 0000 a 0xBFFF FFFF. De manera similar, el bit 1 (FAEHF) en el registro STATUS0 se establece cuando el bit 30 de uno de los registros xFWATTHR cambia, lo que significa que uno de estos registros está medio lleno. La configuración del bit 6 (RSTREAD) del registro LCYCMODE habilita una lectura con restablecimiento para todos los registros de acumulación de vatios-hora, es decir, los registros se restablecen a 0 después de una operación de lectura.

1.9.1.13 Cálculo de la potencia reactiva

El ADE7880 calcula la potencia reactiva fundamental, la potencia determinada solo por los componentes fundamentales de los voltajes y corrientes. También calcula las potencias reactivas armónicas, las potencias reactivas determinadas por los componentes armónicos de las tensiones y corrientes. Una carga que contiene un elemento reactivo (inductor o condensador) produce una diferencia de fase entre el voltaje de CA aplicado y la corriente resultante. La potencia reactiva se define como el producto de las formas de onda de voltaje y corriente cuando todos los componentes armónicos de una de estas señales tienen un desfase de 90°. La potencia reactiva total promedio sobre un número integral de ciclos de línea se calcula según la ecuación 1.17.

$$\boldsymbol{Q = \frac{1}{nT} * \int_{0}^{nT} q(t)dt \ = \sum_{k=1}^{\infty} V_k * I_k * sen(\varphi - \gamma) \quad Ecuación\ 1.17}$$

Dónde:

T es el período del ciclo de línea.

Q se conoce como la potencia reactiva total.

Esta es la relación utilizada para calcular la potencia reactiva total para cada fase. La señal de potencia reactiva instantánea, q(t), se genera multiplicando cada armónico de las señales de voltaje por el correspondiente armónico desfasado 90 ° de la corriente en cada fase. La expresión de la potencia reactiva fundamental se obtiene de la ecuación 1.18 con k = 1, como sigue:

$$\boldsymbol{Q_{fundamental} = V_1 * I_1 * sen(\varphi_1 - \gamma_1)} \quad \textbf{Ecuación 1.18}$$

El ADE7880 calcula la potencia reactiva fundamental utilizando un algoritmo patentado que requiere alguna función de inicialización de la frecuencia de la red y el voltaje nominal medido en el canal de voltaje. Estas inicializaciones se introducen según se describe en la sección 1.9.1.12.1 y son comunes tanto para las potencias fundamentales activas como reactivas.

1.9.1.13.3 Cálculo de energía reactiva

La energía reactiva fundamental se define como la integral de la potencia reactiva fundamental. Similar a la potencia activa, el ADE7880 calcula la integración de la señal de potencia reactiva en dos etapas tal como se muestra en la figura 27.

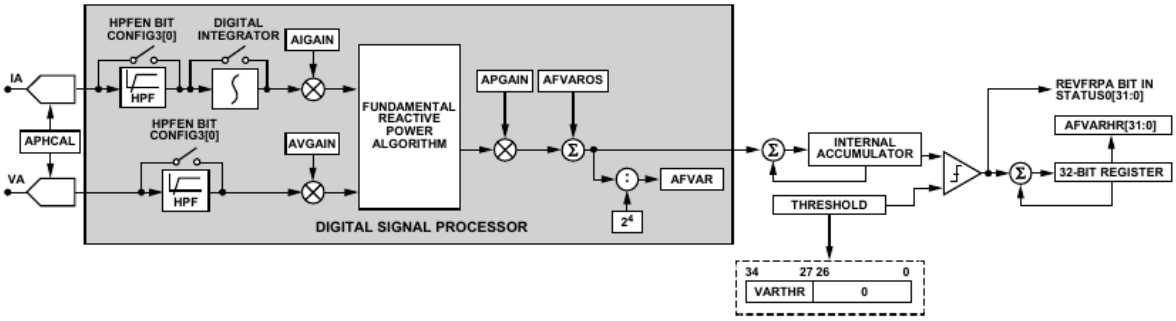


Figura 27: Acumulación de energía reactiva fundamental.

La primera etapa acumula la potencia reactiva fundamental de fase instantánea a 1.024 MHz, aunque son calculadas por el DSP a una tasa de 8 kHz. Cada vez que se alcanza un umbral, se genera un pulso y el umbral se resta del registro interno. El cálculo de la energía en este momento considera el signo de la potencia reactiva. La segunda etapa consiste en acumular los pulsos generados después de la primera etapa en registros internos de acumulación de 32 bits. El contenido de estos registros se transfiere a los registros var-hour (xFVARHR) cuando se accede a estos registros. AFWATTHR, BFWATTHR y CFWATTHR representan energías reactivas fundamentales de fase.

1.9.1.14 Cálculo de potencia aparente

La potencia aparente se define como la potencia máxima que se puede entregar a una carga. Una forma de obtener la potencia aparente es multiplicando el valor rms del voltaje por el valor rms de corriente como se muestra en la ecuación 1.19.

$$\boldsymbol{S = V_{rms} * I_{rms} \quad Ecuación\ 1.19}$$

Dónde:

S = A la potencia aparente.

Vrms e Irms son el voltaje y la corriente rms, respectivamente.

El ADE7880 calcula la potencia aparente en cada fase. La figura 28 muestra la trayectoria del procesamiento de la señal en cada fase para el cálculo de la potencia aparente. Debido a que Vrms e Irms contienen toda la información armónica, la potencia aparente calculada es la potencia aparente total. El ADE7880 calcula las potencias aparentes fundamentales y armónicas determinadas por las componentes fundamentales y armónicas de los voltajes y corrientes.

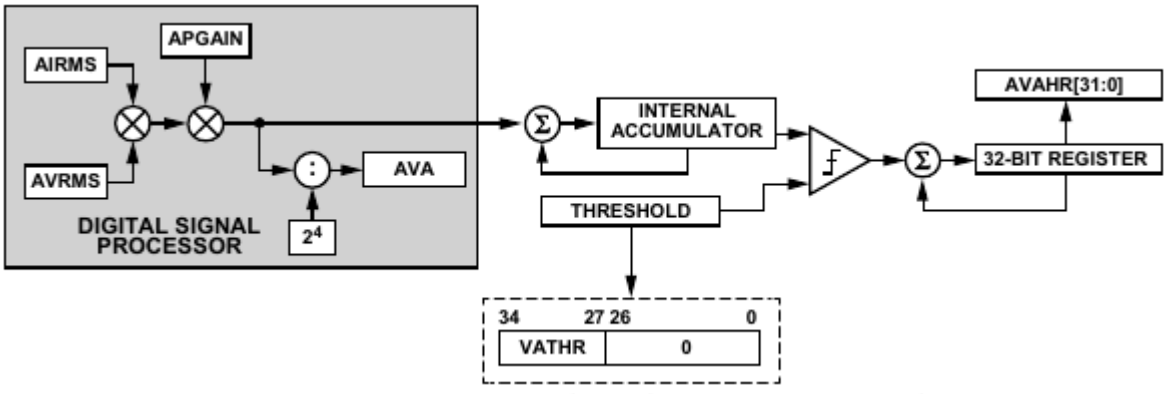


Figura 28: Trayectoria para la potencia y acumulación de energía aparente.

Los valores rms de voltaje y corriente se multiplican juntos en el procesamiento de la señal de potencia aparente. El ADE7880 almacena las potencias aparentes de fase en los registros AVA, BVA y CVA. Los registros de ganancia xPGAIN son usados también para calibrar la ganancia en los cálculos de la potencia aparente, por lo que el uso de estos registros necesita de mucho cuidado debido a que todas las potencia comparten los valores que dichos registros contienen para mayor detalles véase la sección 1.9.1.12.2.

1.9.1.14.1 Cálculo de energía aparente

La energía aparente se define como la integral de la potencia aparente. Similar a las potencias activa y reactiva, el ADE7880 hace la integración de la señal de potencia aparente en dos etapas, como se muestra en la figura 28. La primera etapa acumula la potencia aparente instantánea a una tasa de 8[kHz]. Cada vez que se alcanza el umbral, se genera un pulso y el umbral se resta del registro interno. La segunda etapa consiste en acumular los pulsos generados después de la primera etapa en registros internos de acumulación de 32 bits. El contenido de estos registros se transfiere a los registros xVAHR, cuando se accede a estos registros. La figura 28 muestra este proceso. El umbral está formado por el registro sin signo VATHR de 8 bits concatenado a 27 bits iguales a 0 y es común para todas las potencias totales activas y fundamentales de fase. Su valor depende de cuánta energía se asigne a un LSB. Los registros xVATHR.

1.9.1.15 Cálculo del factor de potencia

El ADE7880 proporciona una medición directa del factor de potencia simultáneamente en todas las fases. El factor de potencia en un circuito de CA se define como la relación entre la potencia activa total que fluye hacia la carga y la potencia aparente. La medición del factor de potencia absoluto se define en términos del adelanto o retraso en referencia a si la corriente, esta adelanta o atrasa con respecto a la forma de onda del voltaje. Cuando la señal de corriente adelanta a la señal de voltaje, la carga es capacitiva y esto se define como un factor de potencia negativo. Cuando la señal de corriente está retrasada con respecto a la señal de voltaje, la carga es inductiva que se define como un factor de potencia positivo. La relación de atraso o adelanto de la señal de corriente con respecto de la señal de voltaje se muestra en la figura 29.

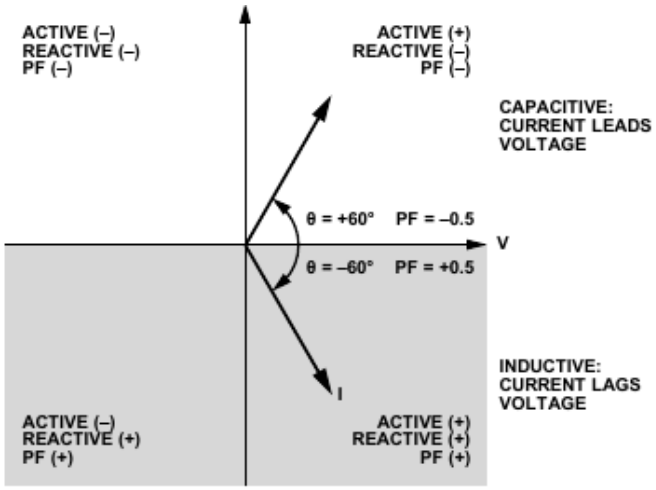


Figura 29: FP para cargas capacitivas e inductivas.

Como se mencionó anteriormente, el ADE7880 proporciona una medición del factor de potencia en todas las fases simultáneamente. Estas lecturas se proporcionan en tres registros firmados de 16 bits, los cuales son APF para la Fase A, BPF para la Fase B y CPF para la Fase C. Los registros están firmados son registros en complemento a dos con el MSB indicando la polaridad del factor de potencia. Cada LSB de los registros APF, BPF y CPF equivale a un peso de 2^{-15}, por lo que el valor de registro máximo es de 0x7FFF y equivale a un valor de factor de potencia de 1. El valor de registro mínimo es de 0x8000 y corresponde a un factor de potencia de -1.

1.9.1.16 Motor de cálculos armónicos

El ADE7880 contiene un motor de cálculos armónicos que analiza una fase a la vez. La información armónica se calcula en una banda de paso sin atenuación de 2,8[kHz] correspondiente a un ancho de banda de −3[dB] de 3,3[kHz] y se especifica para frecuencias de línea entre 45 Hz y 66 Hz. Es esencial que los registros VLEVEL y SELFREQ estén configurados correctamente para obtener mediciones de armónicos confiables. Además, la señal de voltaje debe estar por encima de ± 100[mVpico] para que el motor armónico funcione correctamente. La corriente del neutro también se puede analizar simultáneamente con la suma de las corrientes de fase. La figura 30 presenta un diagrama sintetizado del motor de armónicos, sus configuraciones y sus registros de salida.

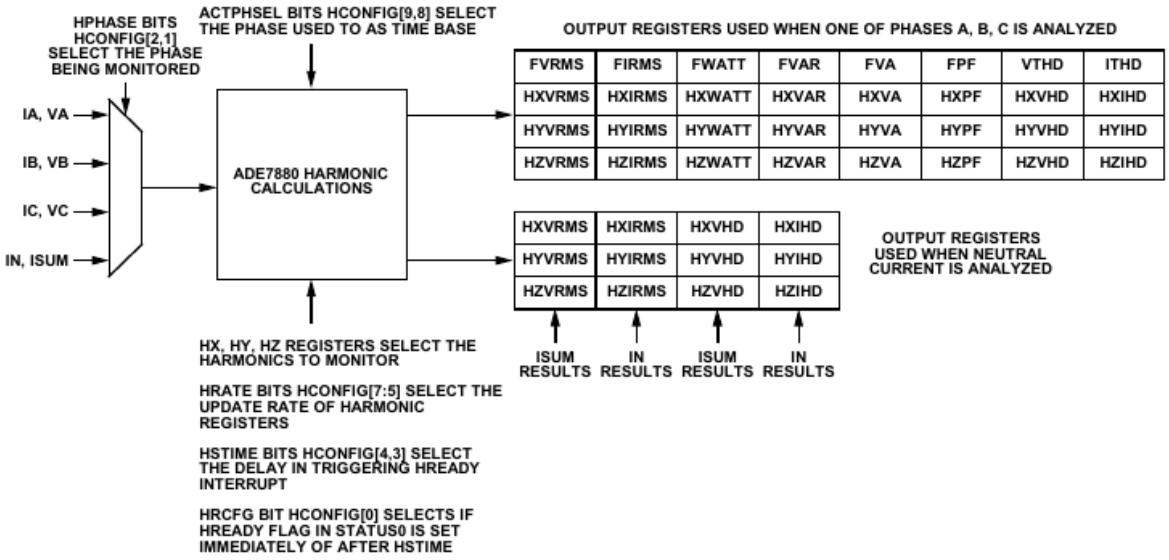


Figura 30: Diagrama en bloque del motor armónico ADE7880.

Los cálculos de armónicos ADE7880 se especifican para frecuencias de línea entre 45[Hz] y 66[Hz]. El número de armónicos N que se pueden analizar dentro de la banda de paso de 2,8[kHz] es el número entero de 2800/f. El número máximo absoluto de armónicos aceptados por el ADE7880 es 63. Cuando se analiza una fase, se calculan los siguientes parámetros:

- Corriente de fase fundamental rms I1.
- Voltaje de fase fundamental rms V1.
- El valor rms de 3 componentes armónicos de corriente de fase Ix, Iy, Iz.
- El valor rms de 3 componentes armónicos de voltaje de fase Vx,Vy, Vz.
- Las potencias fundamentales de fase P1, Q1 y S1.
- Factor de potencia fundamental de fase FP1
- Potencia activa de 3 componentes armónicos de fase Px,Py,Pz.
- Potencia reactiva de 3 componentes armónicos de fase Qx, Qy, Qz.
- Potencia aparente de 3 componentes armónicos de fase Sx, Sy, Sz.
- Factor de potencia de 3 componentes armónicos FPx, FPy, FPz
- THD de la señal de corriente de fase.
- THD de la señal de voltaje de fase.
- Distorsión armónica de 3 componentes armónicos en la fase de corriente.
- Distorsión armónica de 3 componentes armónicos en la fase de voltaje.

1.9.1.16.1 Configuración de los cálculos armónicos

El ADE7880 requiere una base de tiempo proporcionada por un voltaje de fase. El bit 9 y bit 8 (ACTPHSEL) del registro HCONFIG seleccionan este voltaje de fase. Si ACTPHSEL = 00, se utiliza la fase A. Si ACTPHSEL = 01, se usa la Fase B y si ACTPHSEL = 10, se usa la Fase C. Si la tensión de fase utilizada como base de tiempo presenta irregularidades, puede seleccionar otra fase y el motor de armónicos seguirá funcionando correctamente. La fase bajo análisis es seleccionada por los bit 2 y bit 1 (HPHASE) del registro HCONFIG. Si HPHASE = 00, se monitorea la Fase A. Si HPHASE = 01, se monitorea la Fase B y si HPHASE = 10, se monitorea la Fase C. Si HPHASE = 11, se monitorea la corriente de neutro junto con la suma de las corrientes de fase representadas en el registro ISUM.

1.9.1.16.2 Cálculos armónicos cuando se monitorea una fase

Cuando se monitorea una fase, se calcula la información fundamental junto con la información sobre hasta tres armónicos. Los índices de los tres armónicos adicionales monitoreados simultáneamente por el ADE7880 son proporcionados por los registros de 8 bits HX, HY y HZ. Simplemente escriba el índice del armónico en el registro para que ese armónico sea monitoreado. Los componentes fundamentales siempre se monitorean, independientemente de los valores escritos en HX, HY o HZ. El índice máximo permitido en los registros HX, HY y HZ es 63. El valor eficaz de los componentes fundamentales de la tensión de fase y la corriente de fase se almacenan en registros firmados de 24 bits los cuales

son respectivamente FVRMS y FIRMS. La trayectoria de datos asociada se presenta en la figura 31. Similar a las trayectorias de procesamiento de señales de corriente rms y voltaje rms que se presentan en la sección 1.9.1.9, la trayectoria de datos contiene registros de compensación de desplazamiento con signo de 24 bits xIRMSOS, xVRMSOS, donde x = A, B, C para la fase seleccionada. El valor eficaz de la corriente y el voltaje de fase de los tres componentes armónicos se almacenan en registros firmados de 24 bits HXVRMS, HXIRMS, HYVRMS, HYIRMS, HZVRMS y HZIRMS. La trayectoria de procesamiento de señales asociada se presenta en la figura 32 y contiene los siguientes registros de compensación de desplazamiento con signo de 24 bits: HXIRMSOS, HYIRMSOS, HZIRMSOS, HXVRMSOS, HYVRMSOS y HZVRMSOS. Se recomienda dejar los registros de compensación en 0, el valor predeterminado. Las potencias activa, reactiva y aparente del componente fundamental se almacenan en los registros firmados FWATT, FVAR y FVA de 24 bits. La figura 33 presenta la trayectoria de procesamiento de las señales asociadas. Las potencias activa, reactiva y aparente de los 3 componentes armónicos se almacenan en los registros firmados HXWATT, HXVAR, HXVA, HYWATT, HYVAR, HYVA, HZWATT, HZVAR y HZVA de 24 bits. El registro HPGAIN es un registro firmado de 24 bits que se utiliza para escalar la salida de los componentes armónicos de potencia activa y reactiva, como se muestra en la figura 34. Se accede al registro HPGAIN de 24 bits como un registro de 32 bits con los cuatro bits más significativos (MSB) rellenados con 0 y el signo extendido a 28 bits consulte la figura 15 para obtener más detalles. HXWATTOS, HYWATTOS, HZWATTOS, HXVAROS, HYVAROS y HZVAROS son registros de compensación de desplazamiento de 24 bits ubicados en las trayectorias de datos de potencia armónica activa y reactiva. La figura 34 presenta la trayectoria de datos asociada.

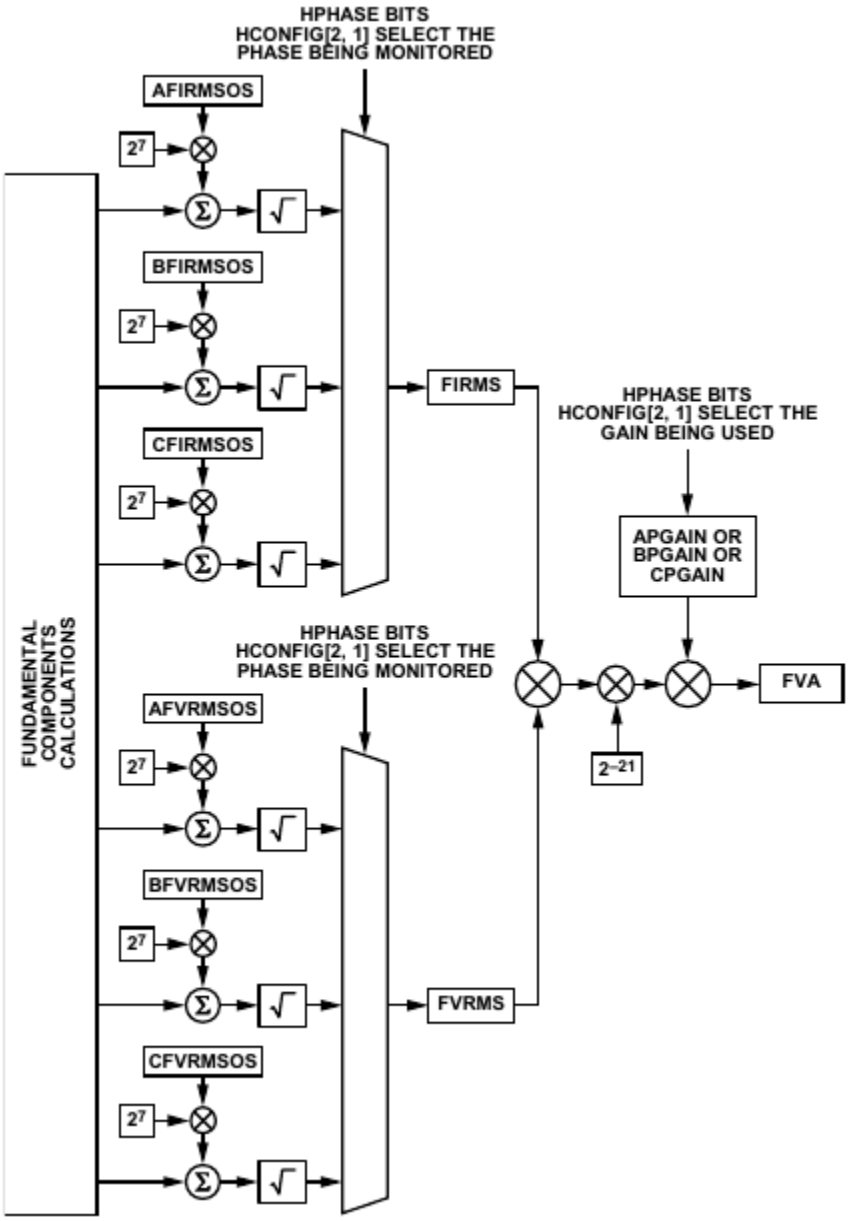


Figura 31: Procesamiento de señales de voltajes y corrientes rms fundamentales.

El factor de potencia del componente fundamental se almacena en el registro firmado FPF de 24 bits. Los factores de potencia de los tres componentes armónicos se almacenan en los registros firmados HXPF, HYPF y HZPF de 24 bits. Las relaciones de distorsión armónica total calculadas usando el valor eficaz de los componentes fundamentales y el valor eficaz de la corriente y el voltaje de fase se almacenan en los registros de 24 bits VTHD e ITHD

Las distorsiones armónicas de los tres componentes armónicos se almacenan en los registros de 24 bits HXVHD, HXIHD, HYVHD, HYIHD, HZVHD y HZIHD.

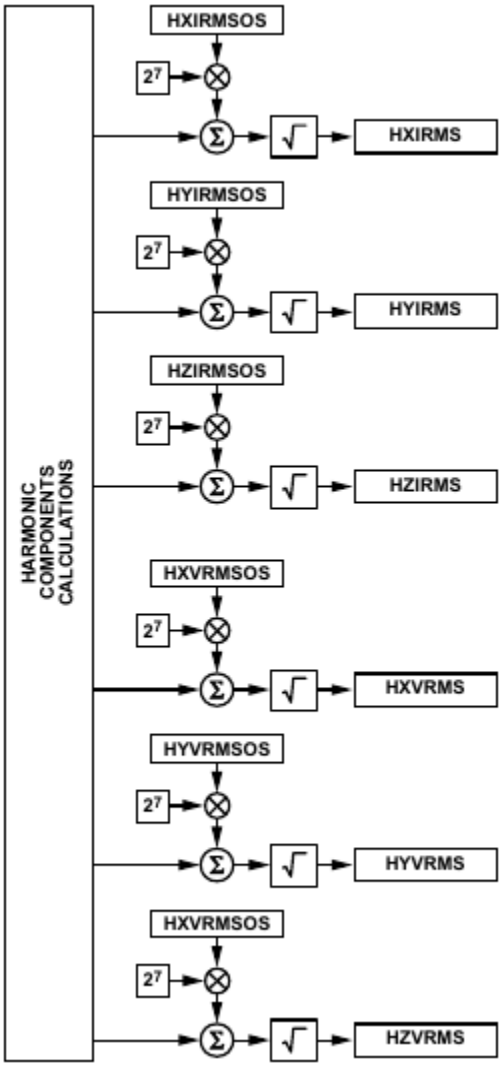


Figura 32: Calculo de las componentes armonicas.

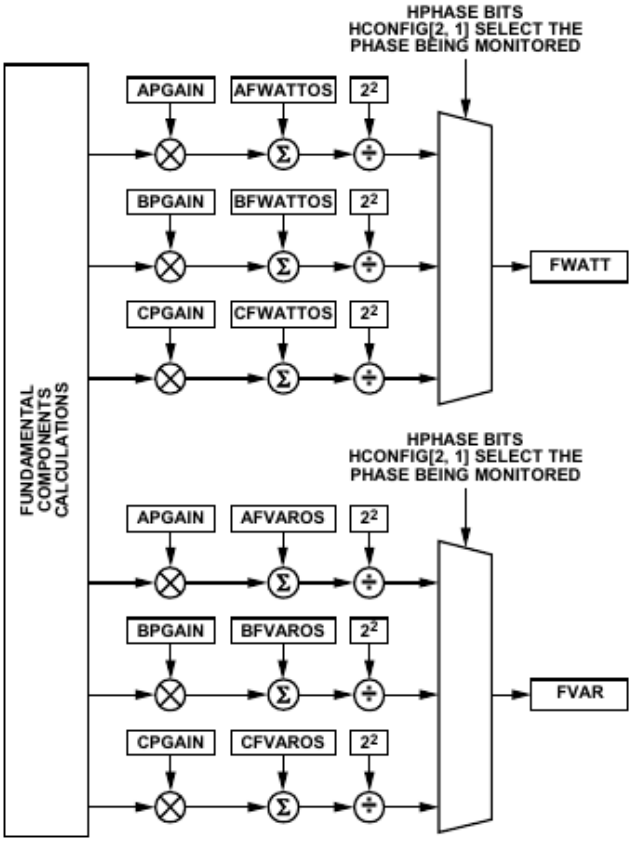


Figura 33: Potencias activas y reactivas fundamentales.

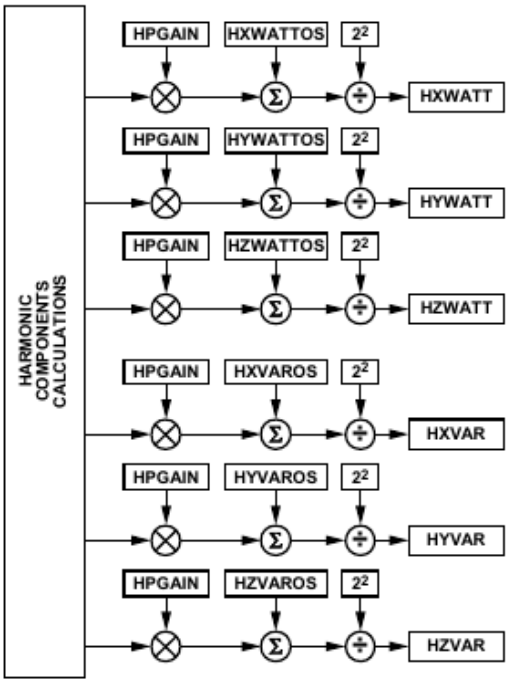


Figura 34: Componentes armónicos potencias activas y reactivas.

1.9.1.16.3 Cálculos armónicos cuando se monitorea el neutro

Cuando se monitorean la corriente neutra y la suma de las corrientes de fase, solo se actualizan los registros relacionados con armónicos de corriente rms. Los registros HX, HY y HZ identifican el índice del armónico, incluido el fundamental. Cuando se analiza una fase, los valores rms fundamentales se calculan continuamente y los resultados se almacenan en registros dedicados FIRMS y FVRMS. Cuando se analiza el neutro, la información fundamental se calcula estableciendo uno de los registros de índice armónico HX, HY o HZ en 1 y los resultados se almacenan en registros armónicos. El índice máximo permitido en los registros HX, HY y HZ es 63. Los registros HXIRMS, HYIRMS y HZIRMS contienen los componentes armónicos rms de la corriente neutra y los registros HXVRMS, HYVRMS y HZVRMS contienen los componentes armónicos rms del registro ISUM. Tenga en cuenta que en este caso, el valor eficaz del componente fundamental no se calcula en los registros FIRMS o FVRMS, pero se calcula si uno de los registros de índice HX, HY y HZ se inicializa con 1. Si el registro HX se inicializa a 1, el ADE7880 calcula las distorsiones armónicas de los otros armónicos identificados en los registros HY y HZ y las almacena en los registros HYVHD, HYIHD, HZVHD y HZIHD de 24 bits. Las distorsiones de la corriente neutra se guardan en los registros HYIHD y HZIHD y las distorsiones del ISUM se almacenan en los registros HYVHD y HZVHD.

1.9.1.16.4 Configuración de la tasa de actualización para los cálculos armónicos

El motor de armónicos funciona a una velocidad de 8[kHz]. Desde el momento en que se inicializa el registro HCONFIG y los índices armónicos se establecen en los registros de índice HX, HY y HZ, los cálculos del ADE7880 tardan típicamente 750 ms en establecerse dentro de los parámetros de especificación. La tasa de actualización de los registros de salida del motor armónico es administrada por los bits [7: 5](HRATE) en el registro HCONFIG y es independiente de la tasa de cálculo del motor. El valor predeterminado es 000 significa que los registros se actualizan cada 125[μs]. Otros períodos de actualización son: 250[μs](HRATE = 001), 1[ms](010), 16[ms](011), 128[ms] (100), 512[ms](101), 1.024[segundos](110). Si los bits de HRATE son 111, los cálculos de armónicos están desactivados.

1.9.1.17 Conversión de energía a frecuencia

El ADE7880 proporciona tres pines de salida de frecuencia: CF1, CF2 y CF3. El pin CF2 se multiplexa con el pin HREADY del bloque de cálculo de armónicos. Cuando HREADY está habilitado, la funcionalidad CF2 está deshabilitada en el pin. El pin CF3 se multiplexa con el pin HSCLK de la interfaz HSDC. Cuando HSDC está habilitado, la funcionalidad CF3 está

deshabilitada en el pin. El pin CF1 y el pin CF2 están siempre disponibles. Después de la calibración inicial. El fabricante o el cliente final verifican la calibración del medidor de energía. Una forma conveniente de verificar la calibración del medidor es proporcionar una frecuencia de salida proporcional a las potencias activa, reactiva o aparente en condiciones de carga constante. Esta frecuencia de salida puede proporcionar una interfaz simple, de un solo cable y aislada ópticamente para el equipo de calibración externo. La figura 35 muestra el proceso de conversión de energía a frecuencia en el ADE7880.

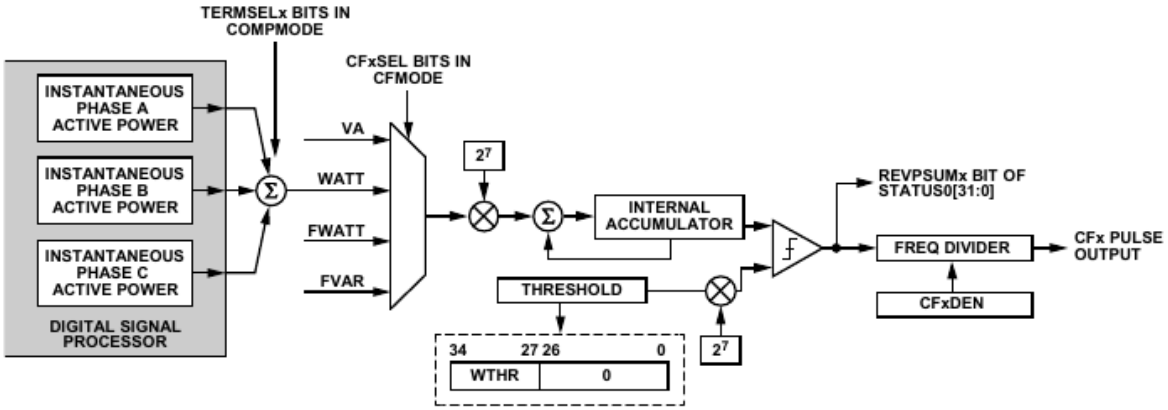


Figura 35: Convertidor de energía a frecuencia.

El DSP calcula los valores instantáneos de todas las potencias de fase: total activo, fundamental activo, fundamental reactivo y aparente. El proceso en el que la energía se acumula en varios registros xWATTHR, xFVARHR y xVAHR ya se ha descrito en las secciones de cálculo de energía: Cálculo de energía activa, Cálculo de energía reactiva fundamental y Cálculo de energía aparente. En el proceso de conversión de energía a frecuencia, las potencias instantáneas generan señales en los pines de salida de frecuencia CF1, CF2 y CF3. Se utiliza un convertidor de energía a frecuencia para cada pin CFx. Cada convertidor suma determinadas potencias de fase y genera una señal proporcional a la suma. Dos conjuntos de bits deciden qué potencias se convierten primero, los bits [2: 0] TERMSEL1 [2: 0], los bits [5: 3] TERMSEL2 [2: 0] y los bits [8: 6] TERMSEL3 [2: 0] del registro COMPMODE. En el registro se configura qué fases, o qué combinación de fases, se agregan. Los bits TERMSEL1 se refieren al pin CF1, los bits TERMSEL2 se refieren al pin CF2 y los bits TERMSEL3 se refieren al pin CF3. Los bits TERMSELx [0] administran la Fase A, los bits TERMSELx [1] administran la Fase B, y los bits TERMSELx [2] administran la Fase C. Establecer todos los bits TERMSELx en 1 significa que todas las potencias trifásicas se agregan en el convertidor CFx. Borrar todos los bits TERMSELx a 0 significa que no se agrega potencia de fase y no se genera ningún pulso CF. Segundo, los bits [2: 0] CF1SEL [2: 0], los bits [5: 3] CF2SEL [2: 0] y los bits [8: 6] CF3SEL [2: 0] en el registro

CFMODE. En este registro se configura qué tipo de energía se usa en las entradas de los convertidores CF1, CF2 y CF3, respectivamente. En la tabla 6 se muestran los valores que puede tener CFxSEL: energías activas totales, aparentes, activas fundamentales o reactivas fundamentales.

CFxSEL	Descripción	Registros con CFxLATCH=1
000	Señal CFx proporcional a la suma de las energías activas de fase totales	AWATTHR, BWATTHR, CWATTHR
001	Reservado	
010	Señal CFx proporcional a la suma de las energías aparentes de fase	AVAHR, BVAHR, CVAHR
011	Señal CFx proporcional a la suma de las energías activas de fase fundamentales	AFWATTHR, BFWATTHR, CFWATTHR
100	Señal CFx proporcional a la suma de las energías reactivas de fase fundamental	AFVARHR, BFVARHR, CFVARHR
101 to 111	Reservado	Registros con CFxLATCH=1

Tabla 5: Configuración de los bits cfxsel del registro cfmode.

De forma predeterminada, los bits TERMSELx son todos 1 y los bits CF1SEL son 000, los bits CF2SEL son 100 y los bits CF3SEL son 010. Esto significa que, de forma predeterminada, el convertidor de energía a frecuencia CF1 produce señales proporcionales a la suma de todas las potencias activas totales trifásicas, CF2 produce señales proporcionales a las potencias reactivas fundamentales, y CF3 produce señales proporcionales a las potencias aparentes. Similar al proceso de acumulación de energía, la conversión de energía a frecuencia se logra en dos etapas. La primera etapa es la misma que se ilustra en las secciones de acumulación de energía de potencias activa, reactiva y aparente (consulte las secciones Cálculo de energía activa, Cálculo de energía reactiva fundamental, Cálculo de energía aparente). La segunda etapa consiste en el divisor de frecuencia por los registros sin firmar CFxDEN de 16 bits. Los valores de CFxDEN dependen de la constante del medidor (CM), medida en impulsos / kWh y de cuánta energía se asigna a un LSB de varios registros de energía: xWATTHR, xFVARHR. El contenido del registro CFxDEN debe ser mayor que 1. Si CFxDEN = 1, entonces el pin CFx permanece activo bajo por solo 1[µs]. Por lo tanto, el registro CFxDEN no debe establecerse en 1. El convertidor de frecuencia no puede admitir resultados fraccionarios; el resultado de la división debe redondearse al número entero más cercano. Si CFxDEN se establece en 0, entonces el ADE7880 lo considera igual a 1. La salida de pulsos CFx está activa a nivel bajo y preferiblemente conectado a un LED, como se muestra en la figura 36.

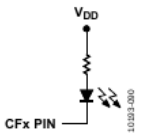


Figura 36: Conexión de los pines CFx.

Los bits [11: 9] CF3DIS, CF2DIS y CF1DIS del registro CFMODE configuran las salidas de los convertidores de frecuencia en los pines CF3, CF2 o CF1. Cuando los bit CFxDIS se establece en 1 el cual es su valor predeterminado, el pin CFx está deshabilitado y el pin permanece alto. Cuando en el bit CFxDIS se escribe un 0, la salida del pin CFx correspondiente genera una señal baja activa lo que significa que está habilitada la salida.

1.9.1.18 Protocolo de comunicaciones I2C

I2C es la abreviatura de circuito ínter-integrado, un protocolo de comunicaciones serial y síncrono que fue desarrollado en los años 1980 por philips semiconductors, su uso inicial fue el poder dotar de comunicación entre múltiples chips en los televisores de la época. Se trata de un protocolo de comunicación muy utilizado en la industria, para establecer intercambio de información entre microprocesadores/microcontroladores y periféricos en sistemas embebidos. A nivel físico el protocolo establece comunicación por medio de dos líneas por las cuales se transmiten las señales de información, estas líneas reciben los siguientes nombres: SCL esta es la línea por donde se transmite la señal de reloj y se utiliza para sincronizar la transferencia de datos a través del bus, SDA es la línea por donde se transmiten los datos, todos los dispositivos que cuenten con este protocolo de comunicación se deben conectar a estas líneas para poder establecer transferencia de información. Las líneas SCL y SDA son del tipo colector o drenador abierto, se polarizan en un estado lógico alto lo que hace necesario la utilización de resistencias conectadas en modo pull-up esto permite una polarización adecuada del bus para poder conectar en paralelo diferentes dispositivos I2C. Sin las resistencias pull-up las líneas SCL y SDA presentarían un nivel lógico bajo en consecuencia no sería posible establecer comunicación.

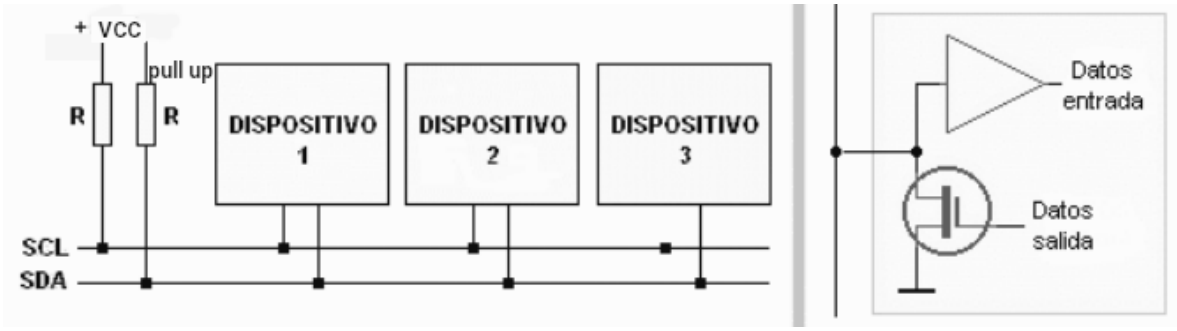


Figura 37: Diagrama general de conexiones I2C.

Los dispositivos conectados al bus I2C cuentan con direcciones (de 7 o 10 bits) únicas que los identifican, de esta forma se puede seleccionar con que dispositivo de entre todos los conectados al bus I2C se desea establecer comunicación ya sea para escritura o lectura. Existen dispositivos en el bus I2C que tienen la característica de iniciar o detener una

comunicación estos reciben el nombre de maestros y además tienen el uso exclusivo de la señal de reloj, a los demás dispositivos conectados al bus se les llama esclavos y solo pueden enviar o recibir información cuando el maestro lo solicite. La información se transmite en mensajes los cuales están compuestos por tramas de datos que son las que transportan las señales de inicio, la dirección del dispositivo, si es lectura/escritura, reconocimiento ACK/NACK, la información del mensaje y la señal de parada, este conjunto de señales forman el mensaje. En la figura 38 se muestra la estructura del mensaje en forma gráfica, para establecer comunicación usando el protocolo de comunicaciones I2C.

Figura 38: Estructura en tramas del mensaje para el protocolo I2C.

Se considera que el bus está libre cuando ambas señales SCL y SDA están en estado lógico alto, en ese estado un dispositivo maestro puede utilizar el bus para generar una condición de inicio, la cual consiste en el cambio de nivel lógico alto a un nivel lógico bajo en la linea SDA mientras la línea SCL se mantiene en estado lógico alto, como se muestra en la figura 39.

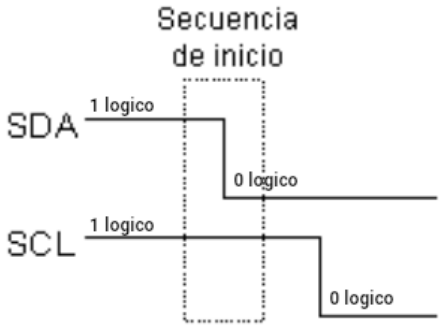


Figura 39: Secuencia de inicio de la comunicación en el bus I2C.

El primer byte transmitido después de la secuencia de inicio contiene siete bits que corresponde a la dirección del dispositivo con el que se establecerá comunicación y un octavo bit que indica la operación que se necesita realizar sea esta de lectura o escritura. Si el dispositivo con el que se necesita establecer comunicación está presente en el bus y en condiciones de establecer la comunicación, este responde con un bit de ACK que consiste en

un bit con estado lógico bajo tras el último transmitido por el maestro. Si todo el procedimiento antes descrito es favorable se considera que la comunicación se ha establecido. La transmisión por la línea SDA inicia con el bit más significativo. Él envió bit a bit por la línea SDA durante la comunicación se realiza cuando la señal SCL se encuentra en un estado lógico bajo. Este proceso se muestra en la figura 40.

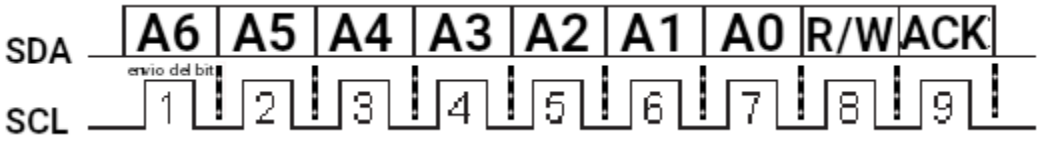


Figura 40: Dirección y asentimiento en tramas de lectura/escritura.

El bit de lectura/escritura(R/W) en un nivel lógico bajo indica que se va a escribir en el dispositivo que tiene la dirección enviada en los bits de A0 – A6. En este caso el maestro envía datos al esclavo y este responde con un asentimiento por cada byte que recibe. Cuando el maestro envía todos los datos de la comunicación procede a liberar el bus de comunicación, para ello tiene que enviar la secuencia de parada, la cual consiste en cambiar el estado lógico en la línea SDA el cual pasa de un estado lógico bajo a un estado lógico alto mientras las línea SCL se mantiene en un nivel lógico alto. La secuencia de parada se muestra en la figura 41.

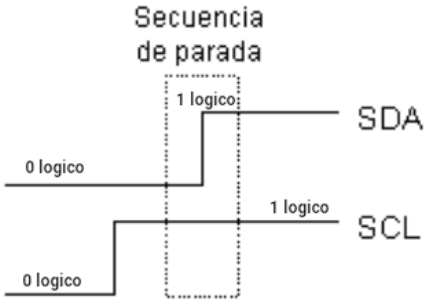


Figura 41: Secuencia de parada de la comunicación en el bus I2C.

Si luego de una transmisión el maestro necesita seguir enviando información debe enviar una nueva secuencia de inicio anulando la secuencia de parada, este proceso recibe el nombre de inicio reiterado, con el cual también puede redireccionar hacia otro dispositivo incluso puede modificar el bit de lectura/escritura. Cuando el bit de lectura/escritura se encuentra en un nivel lógico alto la petición será de lectura. En este caso el maestro toma control de la línea SCL y es el quien genera los pulso de reloj para que el esclavo pueda transmitir la información, tras cada byte recibido el maestro genera una señal de asentimiento (ACK) que indica que la información ha sido recibida con éxito. La operación de lectura tiene cierto

grado de complejidad para realizarla debido a que es necesario indicar el registro que se desea leer del dispositivo esclavo. La operación de lectura se inicia con una escritura previa indicando el registro a leer. Tras la escritura se envía un inicio reiterado y nuevamente la dirección del esclavo pero con el bit R/W en un nivel lógico alto. A partir de ese momento se leen los datos almacenados en el registro seleccionado. Para terminar la comunicación se hace con una secuencia de parada. En las figuras 42 y 43 se muestra el proceso para realizar operaciones de escritura y lectura en registros de 32 bits.

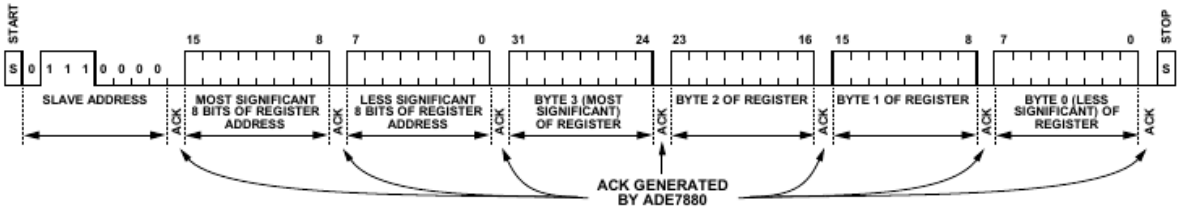


Figura 42: Operación de escritura I2C en un registro de 32 bits.

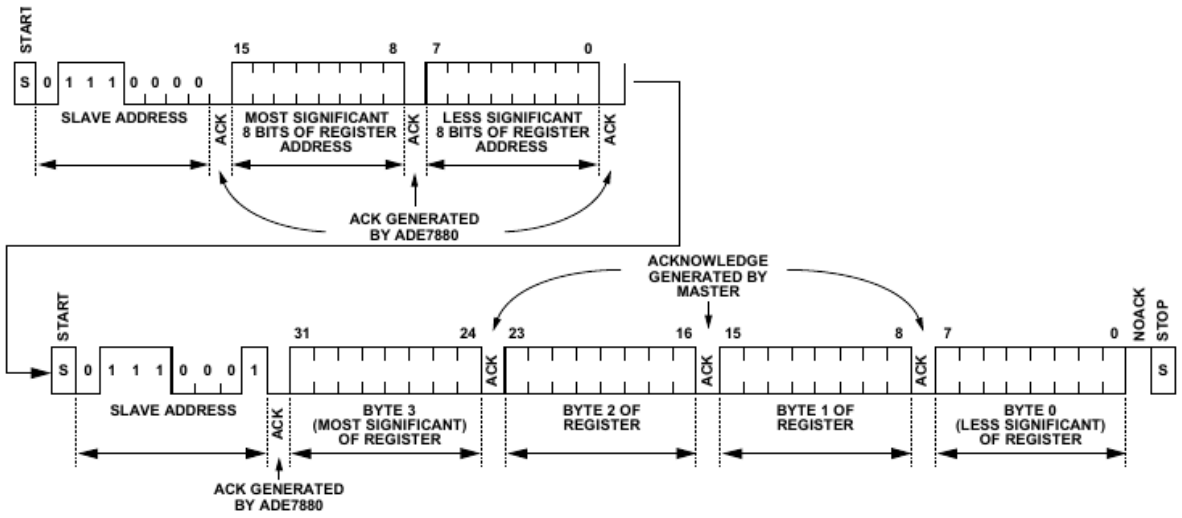


Figura 43: Operación de lectura I2C en un registro de 32 bits.

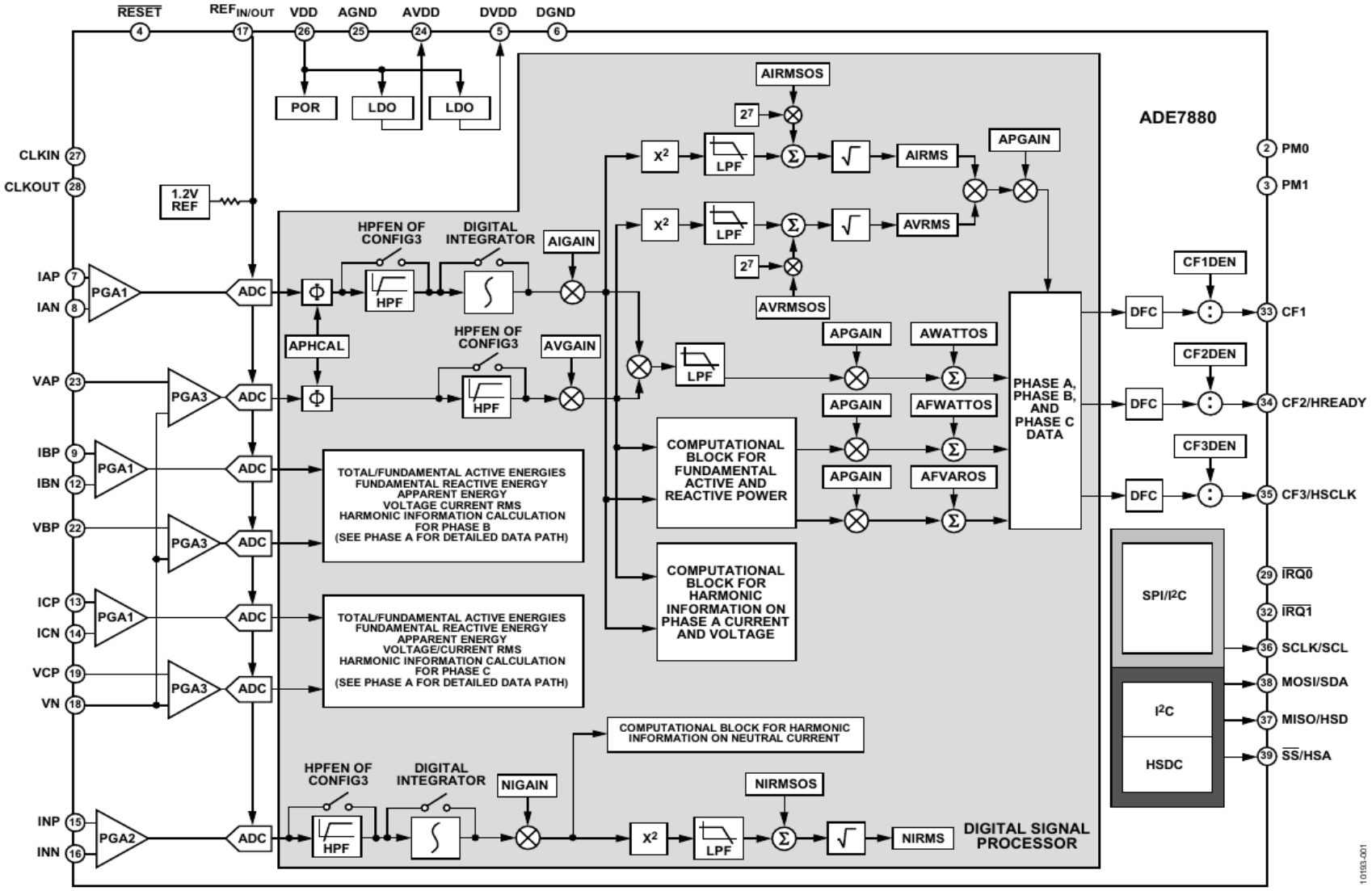


Figura 44: Diagrama en bloques del circuito integrado ADE 7880.

1.10 Raspberry pi ordenador de placa única

La Raspberry Pi es un ordenador de bajo coste y tamaño reducido o SBC desarrollado en el Reino Unido por la raspberry pi Foundation, con el objetivo de potenciar la enseñanza en las ciencias. En el mercado existen diferentes modelos de la placa raspberry pi, para múltiples necesidades estos modelos se pueden adecuar para una gran cantidad de proyectos dependiendo de las características técnicas que estos demanden. Los modelos más conocidos son: raspberry pi A, raspberry pi B y raspberry pi B+, en una evolución desde la 1A hasta la 4B, siendo esta última de los modelos más recientes y robustos en cuanto a hardware por lo que para el presente trabajo se hará uso de la raspberry pi 4B y en la siguiente tabla se muestran las características técnicas más relevantes.

Especificación.	Raspberry pi 4
SOC	BroadcomBCM2711.
CPU	Quad-core Cortex-A72 ARMv8 64bits@1.5GHz.
GPU	Broadcom videocore VI.
Memoria RAM	1GB, 2GB, 4GB.
Almacenamiento	Ranura para memorias microSD.
Salidas de video	Micro HDMI * 2, interfaz DSI para un LCD.
Entradas de video	Conector MIPI CSI para la conexión de una cámara.
Salidas de audio	Conector Jack de 3.5mm, 2 puertos micro HDMI.
USB 2.0	2 Puertos.
USB 3.0	2 Puertos.
Conectividad a la red	Puerto RJ-45- 10/100/1000Mbps, por Hub usb 3.0, wifi 802.11b/g/n/ac de doble banda 2.4/5.0 Ghz, Bluetooth 5.0 BLE.
Puerto GPIO	Pines gpio 17, SPI, I2C, UART.
Juego de instrucciones	RISC de 64 bits.
Consumo de energía	3A, máximo (15.3W).
Fuente de poder	USB-C, por puerto GPIO o por PoE HAT 5V@3A.
S.O soportados	GNU/LINUX, Raspbian

Tabla 6: Especificaciones técnicas generales de la raspberry pi 4.

Para poder hacer uso de la raspberry pi se debe instalar un sistema operativo en una tarjeta microSD, luego de la instalación hay que insertar la tarjeta de memoria microSD en el puerto apropiado de la raspberry pi, esta parte es fundamental puesto que el rendimiento del ordenador desde una perspectiva general depende en gran medida del sistema operativo. Para

este trabajo se ha seleccionado la distribución raspios-buster que es una derivación del sistema operativo Debian de GNU/LINUX, esta distribución raspios-buster esta optimizada para el hardware del ordenador raspberry pi. Los detalles de la instalación se describen en el anexo A. En la figura 45 se describen brevemente los componentes de la raspberry pi 4.

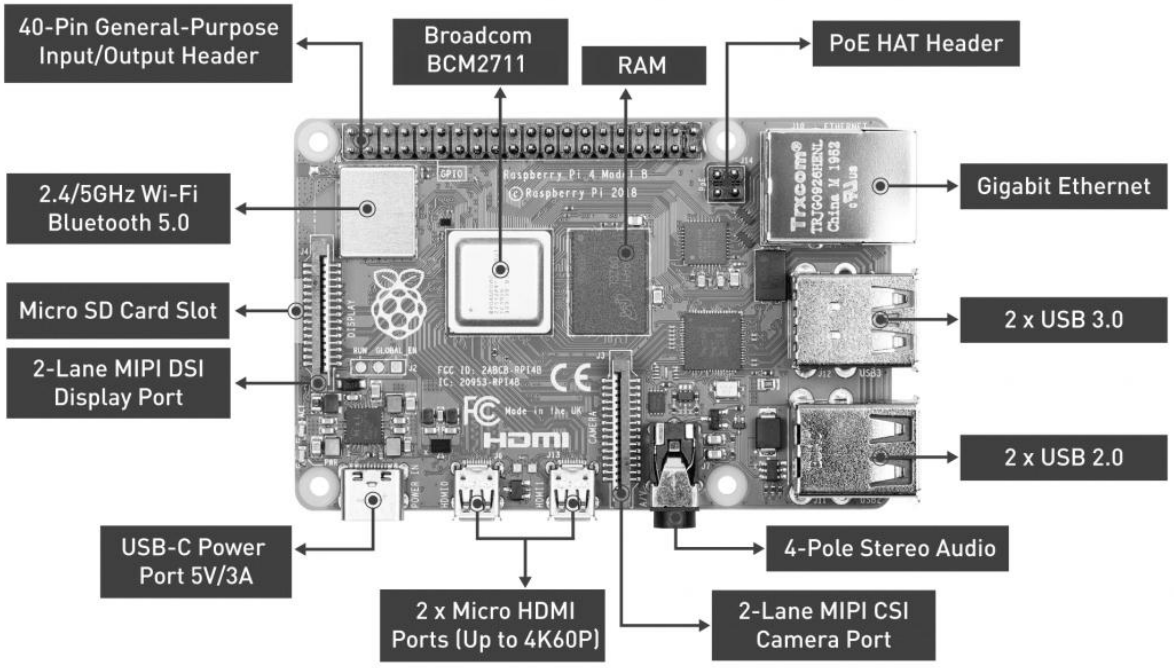


Figura 45: Vista superior del computador de placa única raspberry pi 4.

1.10.1 Puerto de entrada y salida de propósito general (GPIO)

El puerto GPIO es un puerto de entrada y salida de propósito general, es decir consta de una serie de pines o conexiones que pueden ser usadas como entradas o salidas digitales. Este conjunto de pines están incluidos en todas las versiones de las raspberry pi pero con algunas diferencias. Los primeros modelos de esta mini computadora contaban con un puerto GPIO de 26 pines sin embargo en las versiones más recientes el número de pines de conexión aumento a 40, la compatibilidad es completa entre los pines de ambas versiones. El puerto GPIO representa la interfaz entre la raspberry pi y los periféricos que se puedan conectar a ella. Todos los pines del puerto GPIO son del tipo sin búfer esto quiere decir que carecen de protección por lo que se requiere de especial cuidado al efectuar las conexiones para no dañar el puerto. En la figura 46 se muestra la distribución de pines del puerto GPIO y se observa la funcionalidad de cada uno de los pines.

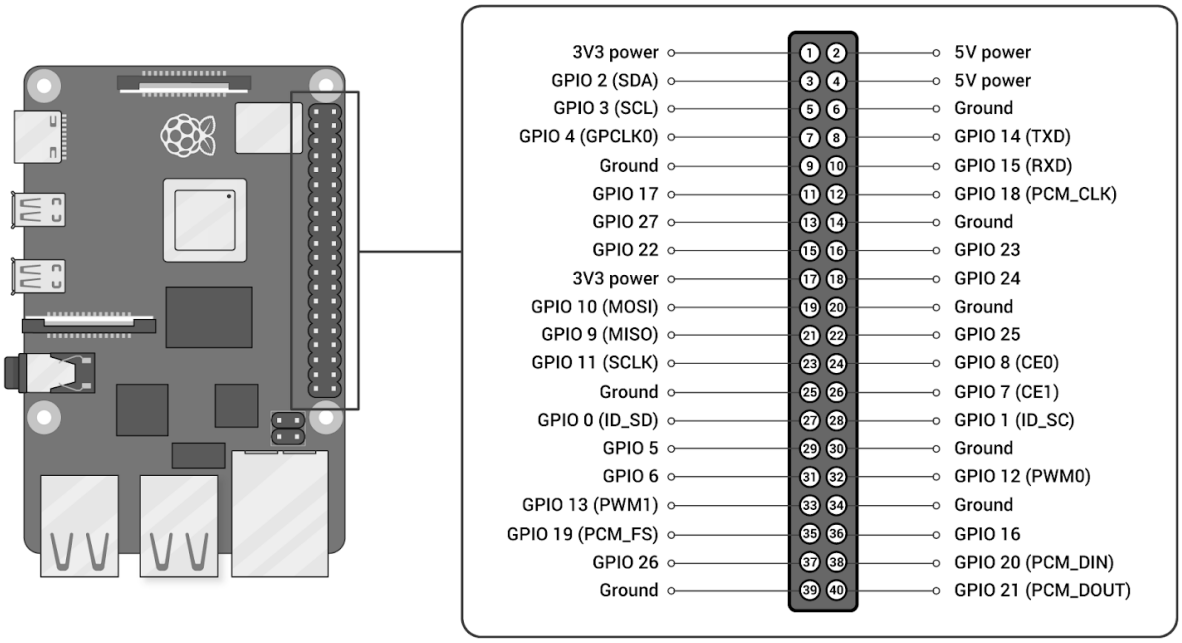


Figura 46: Distribución de pines del puerto GPIO raspberry pi 4.

El puerto GPIO se puede dividir en cuatro secciones que agrupa las diferentes funciones que desarrollan los pines del puerto.

- Pines de alimentación 5[V], 3.3[V] y GND
- Pines NC sin conexión
- Pines configurables son los que se pueden configurar por software como entradas o salidas digitales.
- Pines especiales son los que proporcionan diferentes tipos de interfaces de comunicación entre la raspberry pi y periféricos por ejemplo I2C, SPI, UART, PWM.

1.11 Reloj en tiempo real

Existen en la actualidad una amplia gama de RTC en el mercado. Para el presente trabajo se ha seleccionado el reloj en tiempo real DS3231 que es de alta precisión. Cuenta con un oscilador a cristal con compensación de temperatura este oscilador a cristal viene integrado en el CI DS3231 por lo que la precisión a largo plazo está asegurada. Este RTC mantiene registros de segundos, minutos, horas y fecha (día de la semana, mes y año) la fecha al final del mes se ajusta automáticamente para los meses con menos de 31 días, también cuenta con compensación de la fecha para años bisiestos válida hasta el 2100. El RTC funciona en los formatos de 24h y 12h, cuando se selecciona el formato de 12h presenta un indicador de AM/PM. El DS3231 utiliza un puerto de comunicación I2C para la configuración y

transferencia de datos, las aplicaciones recomendadas para este RTC son: servidores, telemática, gps y medidores de energía. Además se puede destacar su bajo costo y excelente precisión obteniendo una relación costo/beneficio que satisface las necesidades de diseño del presente trabajo. En la figura 4 se muestra el módulo RTC DS3231.

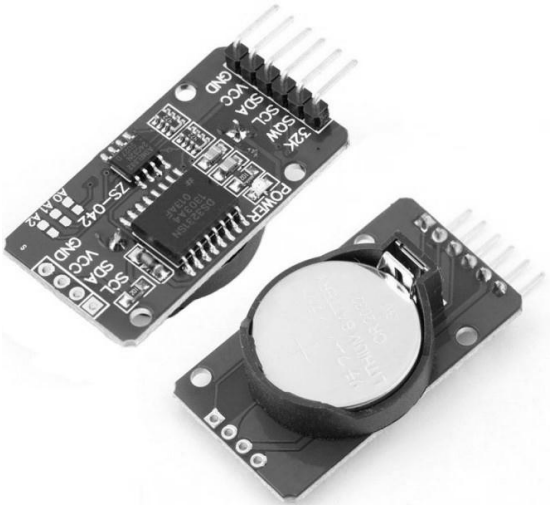


Figura 47: Vistas del módulo RTC DS3231.

1.13 Lenguajes de programación

Es el conjunto de instrucciones a través del cual un programador puede interactuar con las computadoras. Un lenguaje de programación nos permite establecer comunicación con los computadores, teléfonos inteligentes o tabletas, a través de algoritmos escritos con una sintaxis específica para cada lenguaje de programación, estos algoritmos se guardan en archivos con diferentes tipos de extensiones de acuerdo al lenguaje de programación utilizado, por ejemplo para el lenguaje de programación java su extensión es "Nombre del archivo.java", o para el lenguaje de programación C su extensión será "Nombre del archivo.c" estos archivos son procesados por un programa de computadora llamado compilador el cual tiene la función de traducir el algoritmo escrito en un lenguaje de programación determinado y crear otro archivo en otro lenguaje este nuevo archivo contiene lo que se llama lenguaje de maquina o lenguaje de bajo nivel, el contenido interno de este archivo es leído y ejecutado únicamente por el microprocesador de una computadora ya que solo estos dispositivos son capaces de entenderlo. Utilizar un lenguaje de programación requiere estudiarlo y entenderlo desde tres perspectivas las cuales son:

- Sintaxis: es el conjunto de símbolos y reglas para formar sentencias. La sintaxis es la estructura de una declaración en un lenguaje de programación.

- Semántica: son las reglas para transformar sentencias en instrucciones lógicas. La semántica trata sobre el significado de la instrucción. Responde a las preguntas ¿es válida esta instrucción? Si es así, ¿qué significa esta instrucción?, ¿qué se quiere lograr con esta instrucción?
- Pragmática: donde se utilizan las construcciones particulares del lenguaje. La pragmática en programación, se refiere al modo en que el contexto influye en la forma como se interpretan y analizan los problemas que se quieren resolver valiéndose dé un lenguaje de programación.

Existe una gran variedad de lenguajes de programación de los cuales pueden hacer uso los programadores para interactuar con computadores, teléfonos inteligentes o tabletas, esto se logra a través de lo que se conoce como código fuente, todos los lenguajes de programación son diferentes unos de otros aun que comparten las mismas características generales, esta diferencia es muy beneficiosa cuando se tiene que seleccionar un lenguaje de programación para una determinada tarea dentro de la industria. Los lenguajes de programación se utilizan para crear sistemas operativos, programas de escritorio, aplicaciones móviles con la finalidad de resolver problemas o interpretar datos. En el presente trabajo se hace uso del lenguaje de programación java y se utilizaran los IDEs netbeans y android-studio los cuales darán el soporte necesario para el desarrollo del software.

1.13.1 Lenguaje de programación java

Java es un lenguaje de programación y a su vez una plataforma informática que fue comercializada por primera vez en 1995 por la empresa Sun Microsystems. El objetivo de este lenguaje es que los programadores tuvieran que escribir el código de un programa una sola vez y que este, pudiese ejecutarse en cualquier dispositivo computacional. Esto es posible debido a la máquina virtual de java (JVM), que brinda esa portabilidad y que le ha dado una posición muy importante dentro del mundo tecnológico que va desde ordenadores portátiles hasta centros de datos, pasando por las consolas de video juegos, computadores de sobremesa, teléfonos inteligentes y finalmente la internet. Java es un lenguaje de programación orientado a objetos (POO), independiente de la plataforma de hardware donde se desarrolla. Es un lenguaje de programación con una curva de aprendizaje baja lo que indica que es relativamente fácil de aprender. Java como lenguaje de programación es robusto, ofrece un manejo automático de la memoria lo que ayuda a reducir errores. La comunidad de programadores java existente es muy extensa a nivel mundial y muy activa lo que genera una gran cantidad de recursos actualizados. Estructuralmente el lenguaje java comienza con paquetes. Un paquete es el mecanismo de espacio de nombres del lenguaje java. Dentro de los paquetes se encuentran las clases y dentro de las clases se encuentran los métodos, las variables, las constantes, entre otros. Cuando se escribe un programa en lenguaje java se crean archivos con extensión .java que luego al ser compilados por el compilador es

verificada la sintaxis del código contenido en los archivos .java si todo esta correcto el compilador convierte los archivos .java en código byte y este lo escribe en archivos .class. Los códigos byte son instrucciones estándar destinadas a ejecutarse en una JVM. El lenguaje java se creó para cumplir cinco objetivos principales:

- Usar el paradigma de la programación orientada a objetos.
- Permitir la ejecución de un mismo programa en múltiples S.O.
- Incluir por defecto soporte para trabajar en la web.
- Diseñarse para ejecutar código en sistemas remotos de forma segura.
- Ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos.

1.14 Entorno de desarrollo integrado

Es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollador de software. Un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y depuradores, compiladores e intérpretes. Los IDE están diseñados para maximizar la productividad de los programadores proporcionando componentes muy unidos con interfaces de usuario. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software. Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo. Reduciendo tiempo en ajustes, se incrementa la productividad en el desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado. Algunos IDE están dedicados específicamente a un lenguaje de programación, permitiendo que las características sean lo más cercanas al paradigma de programación de dicho lenguaje. Como contraparte existen IDE para múltiples lenguajes de programación.

1.14.1 Entorno de desarrollo integrado netbeans

Netbeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación java. Existen además un número importante de módulos para extenderlo. Netbeans es un producto libre y gratuito sin restricciones de uso. Nerbeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. La empresa Sun Microsystem fundo el proyecto de código abierto en junio del año 2000 y continúa siendo el patrocinador principal de los proyectos. Actualmente Sun Microsystem pasó a ser parte de Oracle Corporation. El IDE netbeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componente de software llamados módulos. Un módulo es un archivo java que contiene clases de java escritas para interactuar

con las APIs de netbeans y un archivo especial (manifest file) que lo identifica como modulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándoles nuevos módulos y esto facilita que otros programadores puedan extenderlas. Netbeans es un framework que simplifica el desarrollo de aplicaciones java. El IDE netbeans ofrece servicios reusables comunes para las aplicaciones de escritorio, permitiendo a los desarrolladores centrarse en la lógica de sus aplicaciones. El IDE netbeans es libre de código abierto y multiplataforma con soporte integrado para el lenguaje de programación java. En la figura 48 se muestra el IDE netbeans en ejecución.

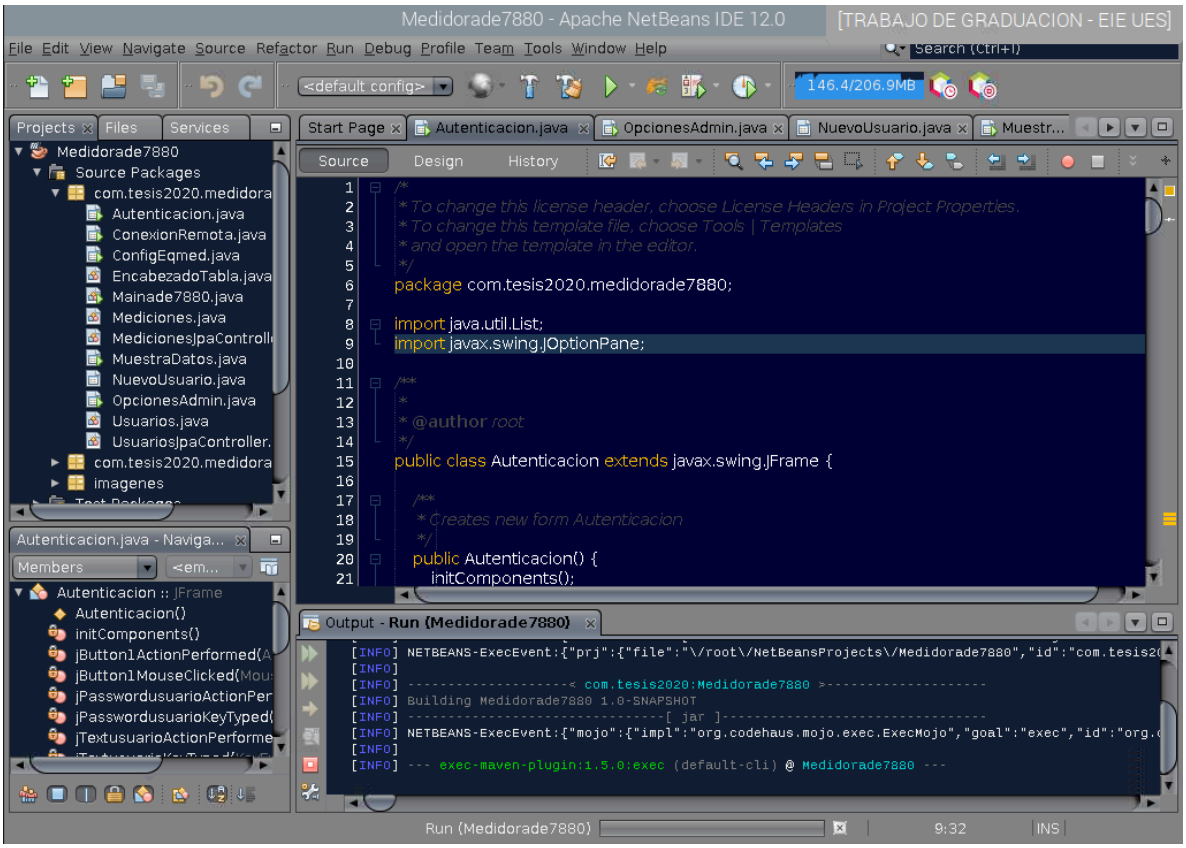


Figura 48: Entorno de desarrollo integrado apache netbeans.

1.14.2 Entorno de desarrollo integrado android-studio

Es el entorno de desarrollo oficial para la plataforma android. Fue anunciado en mayo 2013 y reemplazo al eclipse como IDE oficial para el desarrollo de aplicaciones android. La primera versión estable fue publicada en diciembre del 2014. Este IDE está basado en intelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a traves de la licencia apache 2.0. Está disponible para los sistemas operativos GNU/Linux, MacOs y Microsoft Windows. Ha sido diseñado específicamente para el desarrollo del sistema operativo android. Con cada

versión nueva android-studio incorpora nuevas funcionalidades. Android-studio al igual que muchos otros IDEs es un software que incluye los servicios y herramientas para que un desarrollador de software sea capaz de crear nuevas aplicaciones, también incluye un compilador basado en gradle y soporta los lenguajes de programación C++, Java y Kotlin. En la figura 49 se muestra el IDE android-studio en ejecución.

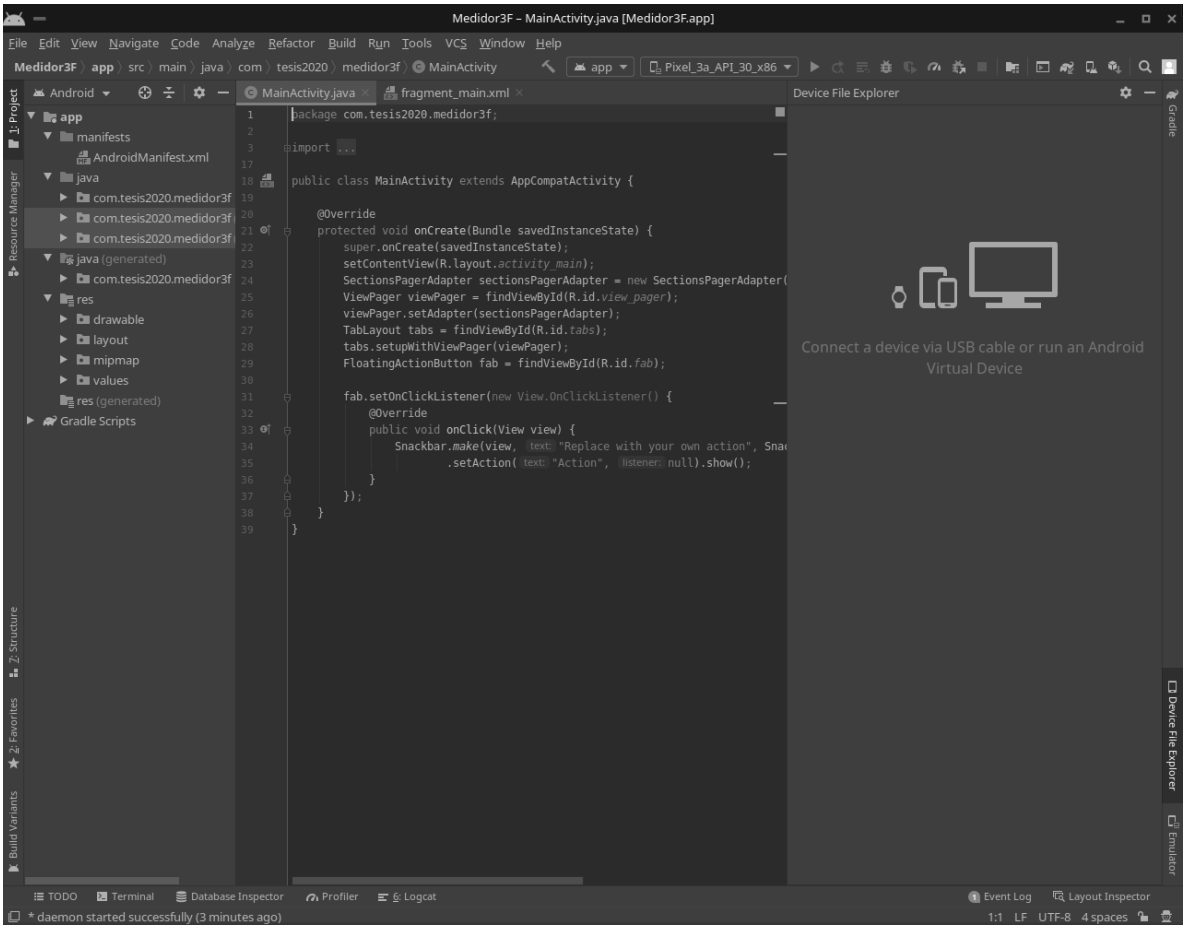


Figura 49: Entorno de desarrollo integrado android-studio.

1.15 Software de diseño asistido por computadora (CAD)

Freecad es un programa de modelado paramétrico 3D desarrollado por Jürgen Riegel, Werner Mayer y Yorik van Havre, freecad es de código abierto y de libre distribución. El programa está actualmente disponible en la versión 0.19. En el sector de la impresión 3D es una herramienta con un nivel de uso elevado, sin embargo las capacidades de este software se extienden más allá del modelado 3D. Freecad admite una amplia gama de formatos de archivos algunos de ellos son: STEP, IGES, STL, SVG, DEX, OBJ, IFC, DAE. Este software está disponible para los sistemas operativos GNU/LINUX, WINDOWS y MAC. Freecad

tiene la característica de ser modular lo que significa que se pueden adicionar funcionalidades o módulos al programa base con lo cual el soporte se vuelve más robusto para el desarrollo de diferentes tipos de trabajos. La primera versión de este software data de octubre del año 2002 y la última versión es de abril 2021. Está disponible en por lo menos 15 idiomas. Freecad es una herramienta que forma parte de los llamados entornos de trabajo en diseño asistido por computadora. Es una herramienta útil para la asistencia en ingeniería y el diseño de elementos mecánicos. Con freecad es posible crear, modificar y diseñar objetos de la vida real de cualquier tamaño. Es el tipo de software que hace posible el diseño de multitud de proyectos y que además se usa en diferentes sectores de la industria, desde el diseño de envases, pasando por el diseño de piezas mecánicas, motores, estructuras de todo tipo, vehículos, circuitos y mucho más. Con la llegada de la impresión 3D, estos programas se han vuelto extremadamente prácticos en los ámbitos industriales y académicos en las áreas de la ingeniería. En la figura 50 se muestra el programa freecad.

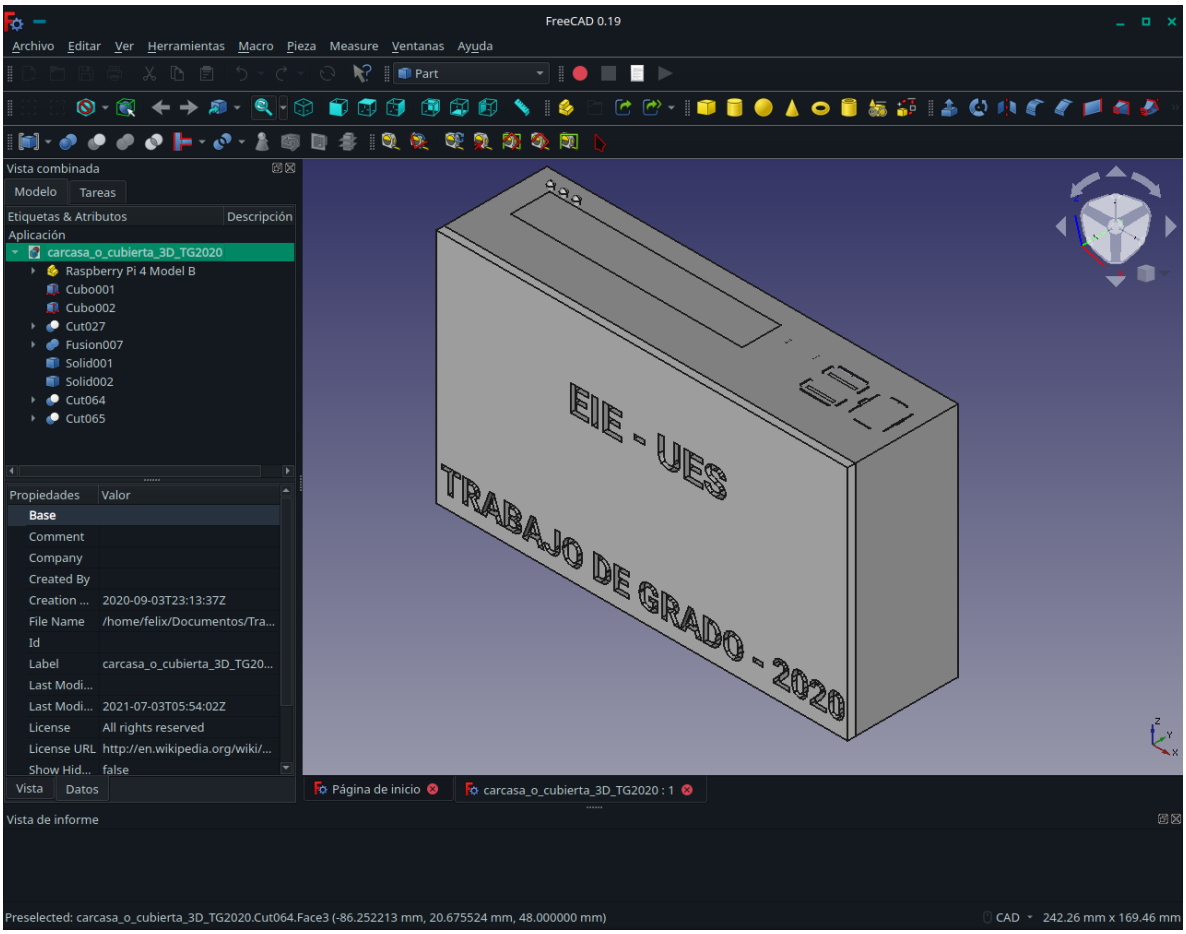


Figura 50: Freecad diseño asistido por computadora.

1.16 Diseño electrónico asistido por computadora (ECAD)

Kicad es una herramienta de software libre para el modelado de circuitos impresos en el desarrollo de hardware. Facilita el diseño esquemático para el desarrollo de circuitos electrónicos y su posterior conversión a placas de circuito impreso (PCB). Kicad consta de una serie de aplicaciones que han sido publicadas con licenciamiento GNU GPL v3 y estas son: Eeschema: editor de esquemas electrónicos, Pcbnew: editor de circuitos impresos, CvPcb: selector de huellas impresas para los componentes usados en el diseño, kicad: manejador de proyectos, entre otras mini aplicaciones útiles para el diseño de un circuito de calidad. Las capacidades actuales para el modelado de circuitos electrónicos, representación 3D y depurado de trazados son bastantes completas. Grupos científicos como la Organización Europea para la Investigación Nuclear o por sus siglas CERN, hacen uso de esta herramienta y al mismo tiempo contribuyen en el desarrollo del mismo. Es un proyecto iniciado en 1992 por JeanPierre Charras y que continúa en desarrollo gracias a una comunidad que se denominan como “KiCadDevelopers Team”. Dicha comunidad está especializada en el área técnica de la electrónica. En las figuras 51, 52, 53 y 54 se muestran las aplicaciones más importantes de kicad.

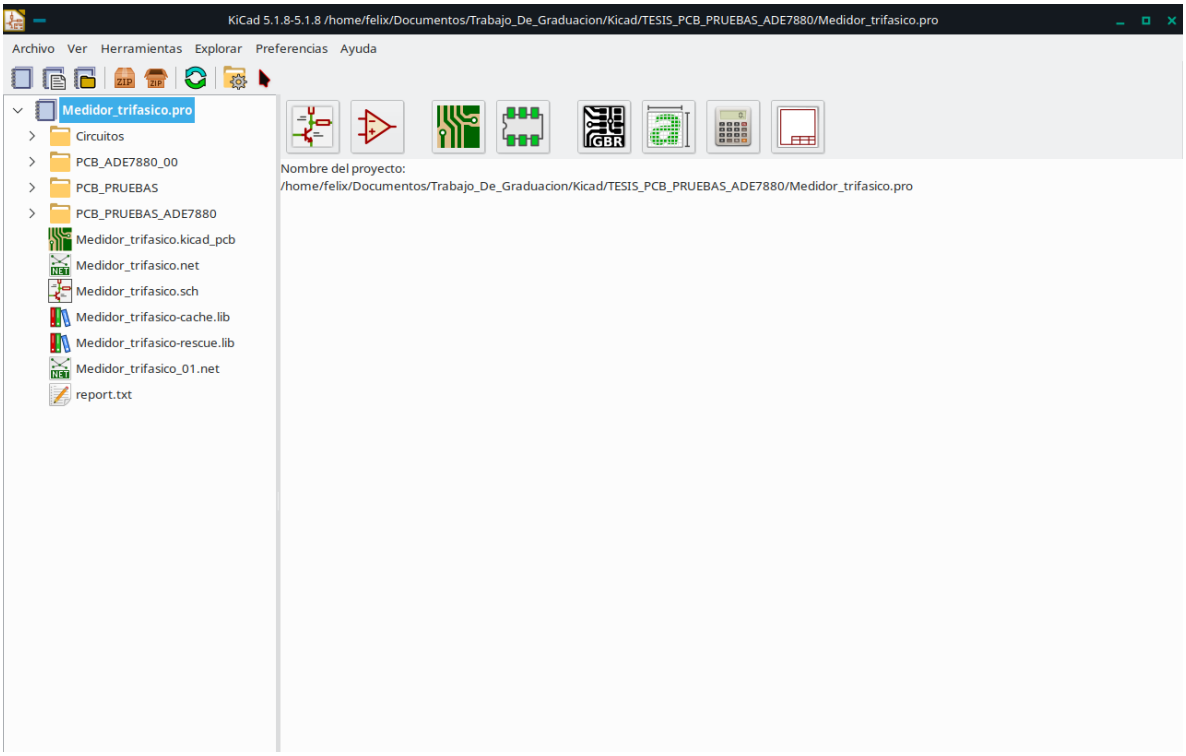


Figura 51: Kicad, sistema integrado de proyectos.

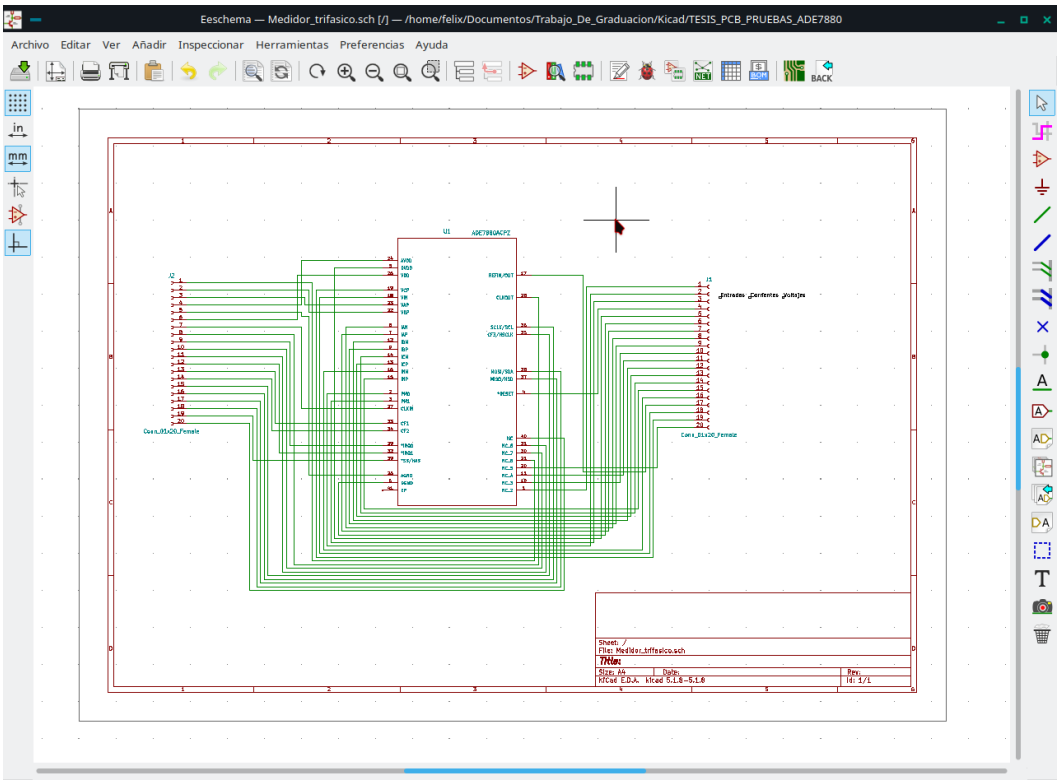


Figura 52: Aplicación Eschema diseño de circuitos electrónicos.

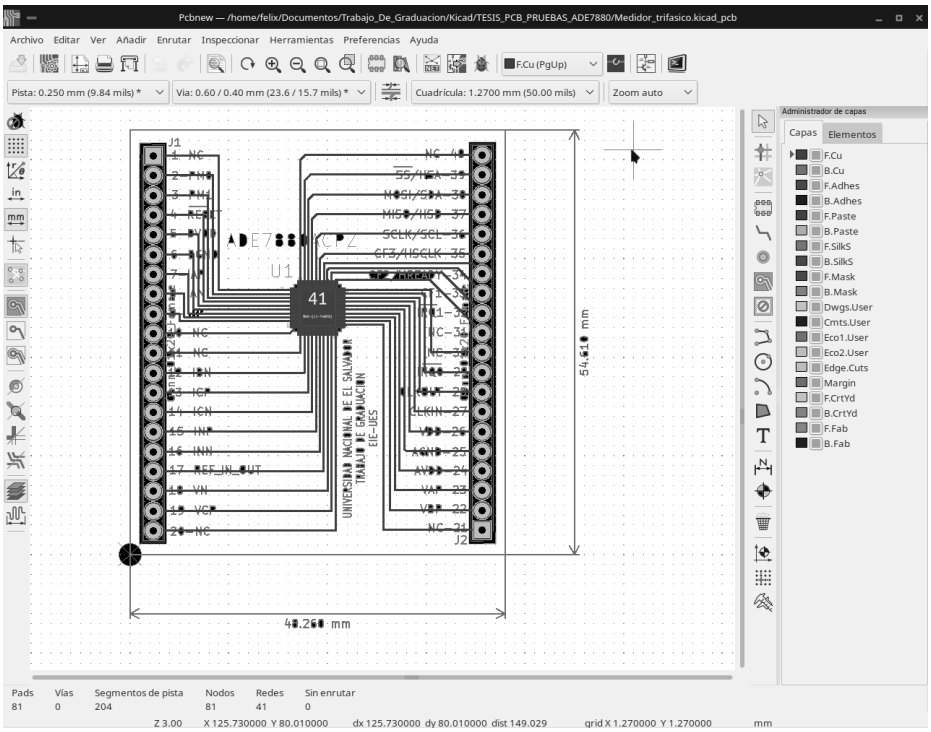


Figura 53: Aplicación pcbnew diseño de circuitos impresos.

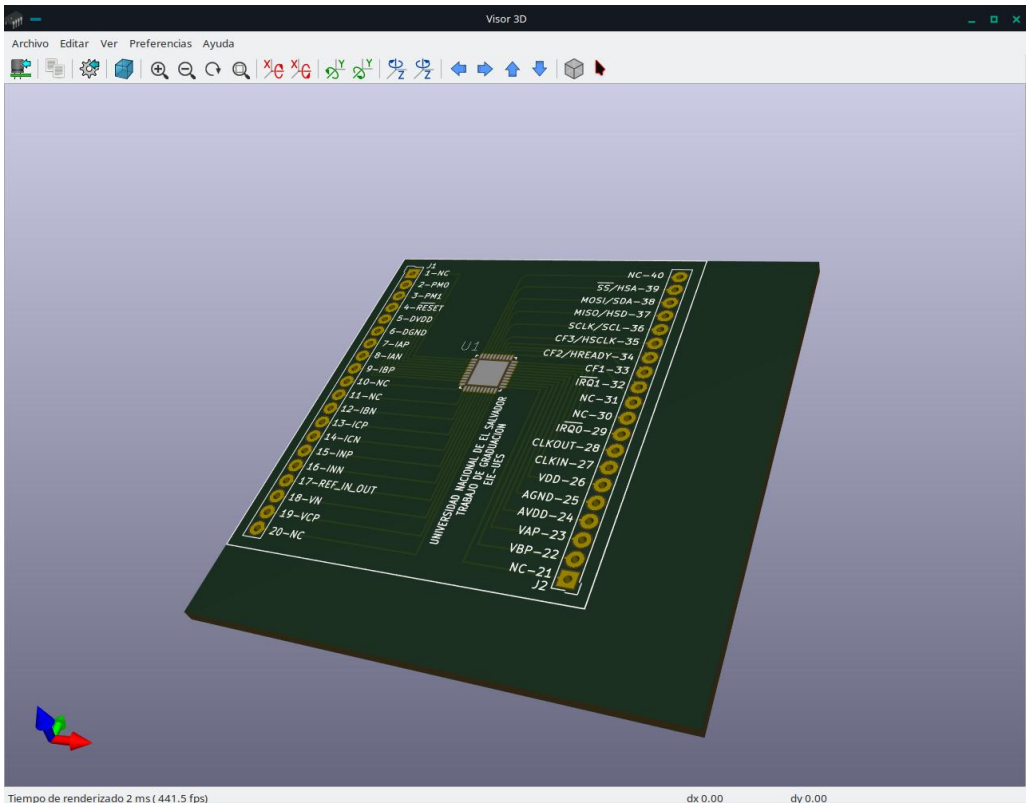


Figura 54: Aplicación visor 3D del diseño de circuito impreso.

1.17 Base de datos relacional

Las bases de datos son parte importante en los actuales sistemas informáticos, puesto que estos sistemas/programas necesitan extraer o almacenar información en tiempo de ejecución y además que esta información sea confiable, esto se hace posible al utilizar un sistema de bases de datos. Las aplicaciones de software posibilitan la interacción del usuario con programas desarrollados para poder poner a disposición una parte de la información o registros específicos dentro de un conjunto de datos almacenados. Una base de datos relacional es un tipo de base de datos la cual almacena y proporciona acceso a los registros que la componen. Las bases de datos relacionales se basan en el modelo relacional una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila de la tabla es un registro llamadas tuplas con un ID único llamado clave, las columnas de las tablas contienen atributos de los datos y cada registro generalmente contiene un valor para cada atributo, lo que facilita las relaciones entre los registros de datos. El modelo relacional creo un estándar en el que se puede representar y consultar datos desde múltiples aplicaciones, unas de las principales fortalezas del modelo es la utilización de tablas, que son una forma intuitiva, eficiente y flexible de almacenar y acceder a información estructurada. El lenguaje de programación que se utiliza para leer o escribir en una base de datos es el

lenguaje de consulta estructurada o SQL. Existen una gran variedad de sistemas de gestión de bases de datos (SGBD) entre los que se pueden mencionar algunos tales como:

- MYSQL: es una SGBD muy utilizada a nivel global.
- MARIADB: es una SGBD derivada de MYSQL, es de código abierto y ha ido creciendo su uso.
- SQLITE: es una biblioteca de programas con licencia de dominio público que contiene un SGBD.

En este trabajo se ha elegido el sistema de gestión de bases de datos a SQLITE.

1.17.1 Sistema de gestión de bases de datos sqlite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en lenguaje C. SQLite es un proyecto de dominio público creado por D.Richard Hipp. A diferencia de los SGBD cliente/servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a traves de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos. El conjunto de la base de datos como definiciones, tablas, indices y los propios datos, son guardados como un solo fichero estándar en la maquina host. Este diseño se logra bloqueando el fichero de base de datos al principio de cada transacción. En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño y también permite la inclusión de campos tipo Blob. La biblioteca implementa parcialmente el estándar SQL-92.

1.18 Servidor de aplicaciones web

Es un programa informático que ejecuta un servicio y desarrolla la función de recibir peticiones provenientes de un cliente/usuario, para luego procesarlas y enviar una respuesta. Las conexiones que se realizan en este proceso pueden ser unidireccionales, bidireccionales, síncronas o asíncronas. Estas se establecen entre el servidor y el cliente. Para la transmisión y recepción de datos se debe utilizar un protocolo de comunicación el generalmente utilizado es HTTP, perteneciente a la capa de aplicación según el modelo OSI.

1.18.1 Servidor de aplicaciones web apache tomcat

Tomcat es un contenedor de servlets que se utiliza en la referencia oficial de la implementación para java servlets y java server pages, las cuales son importantes tecnologías web basadas en java. Apache tomcat es desarrollado bajo el proyecto jakarta en la apache

software foundation. Apache tomcat es un contenedor web con soporte para servlets y JSPs. E incluye el compilador jasper que compila JSPs convirtiéndolas en servlets. Debido a que apache tomcat fue escrito en lenguaje de programación java, funciona en cualquier sistema operativo que disponga de una máquina virtual java (JVM). En la figura 55 se muestra la página principal del servidor apache tomcat.

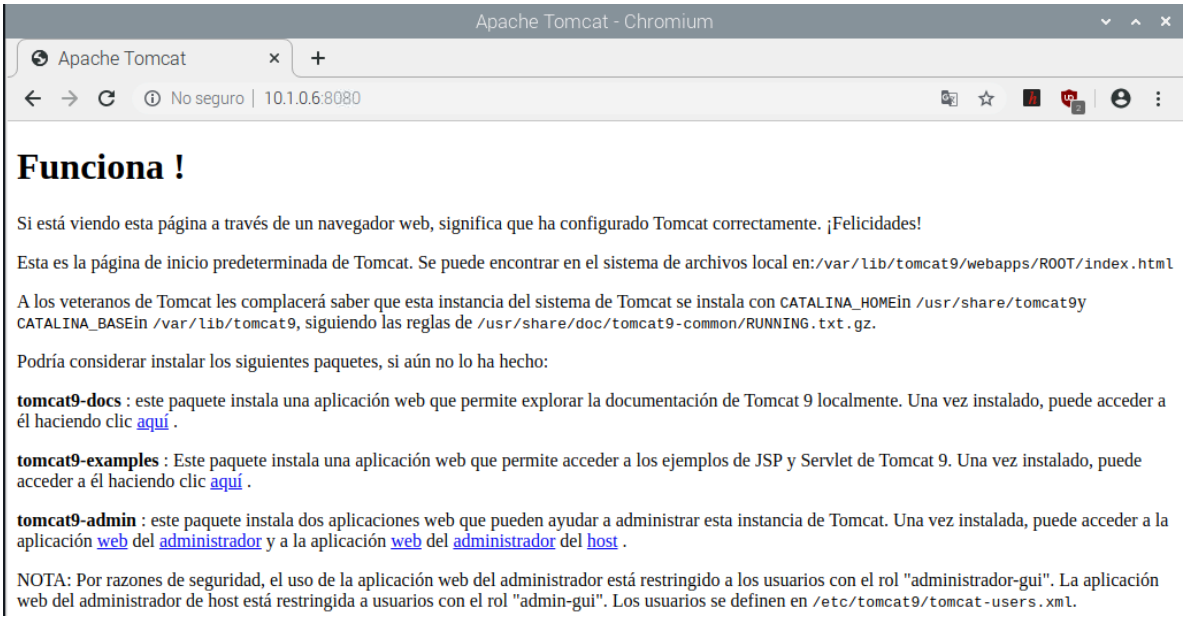


Figura 55: Página principal del servidor apache tomcat.

CAPÍTULO II

DISEÑO, CONSTRUCCIÓN Y DESARROLLO DE LA SOLUCIÓN DEL HARDWARE

2.0 Introducción

En la figura 56 se presenta el diagrama en bloques completo por medio del cual se identifican todas las partes que le dan soporte al medidor trifásico. Se muestra en forma general la interacción del sistema de medición, tanto la parte de hardware como la parte de software. Se parte de un sistema trifásico con sus fases identificadas como A, B, C y el Neutro, un sistema en configuración estrella, luego se muestran las conexiones de las fases del sistema eléctrico con los sensores y acondicionadores de señal estos están identificados por los bloques tipo cajas con las leyendas de cada fase de voltaje, corriente y el neutro según sea el caso, las señales de salida de estos bloques van conectados a los pines de entradas del medidor dedicado ADE7880 estas entradas están conectadas internamente en el CI ADE7880 a los amplificadores de ganancia programable, las salidas de estos van a las entradas de los convertidores analógico a digital sigma-delta las salidas de los ADC van a las entradas del procesador de señales digital. Luego del procesamiento de las señales los resultados se almacenan en los registros de memoria interna de la DSP, para operaciones de lectura. El convertidor de energía a frecuencia utiliza los registros de acumulación que están mapeados en la memoria interna del DSP para proporcionar información por medio del sensor óptico y según la configuración de los registros Cfx puede darnos información de las potencias y energías por fase. Por medio del puerto de comunicaciones I2C se establece comunicación entre el CI ADE7880, el RTC DS3231 y el software de gestión interna o "Demonio-pi4", por medio de este software se realizan las lecturas de los registros en la memoria interna de la DSP, luego por software se convierten los datos que contienen los registros a formato decimal base diez, en este estado los datos son utilizados para determinar las magnitudes de voltajes, corrientes, potencias, factor de potencia, distorsión armónica del voltaje y la corriente. Determinadas las magnitudes de los parámetros eléctricos antes mencionados se procede a almacenarlos en una base de datos por medio del gestor de bases de datos sqlite para su posterior uso ya sea para ser mostrados como registros de mediciones en una tabla administrada por el gestor interno o su representación en forma gráfica. Por medio del protocolo de comunicaciones I2C se configura como reloj del sistema un RTC de alta precisión el cual es DS3231 para tal fin se hace uso de un módulo del kernel llamado RTC_DS1307 este módulo le da soporte a varios modelos de RTC para sistemas operativos GNU/LINUX. Los indicadores en el equipo de medición son controlados por medio del puerto de propósito general GPIO de la raspberry pi 4, por medio de este puerto se controlan cuatro diodos led que indican la operatividad del equipo de medición. El bloque que forma la conexión remota está integrada la aplicación android instalada en un equipo móvil estos dos elementos forman el cliente y en el servidor apache tomcat corre el software desarrollado. Estos dos elementos forman el lado del servidor. Adicionalmente el prototipo de debe integrar a una red tipo LAN por lo que es necesario contar con un router inalámbrico.

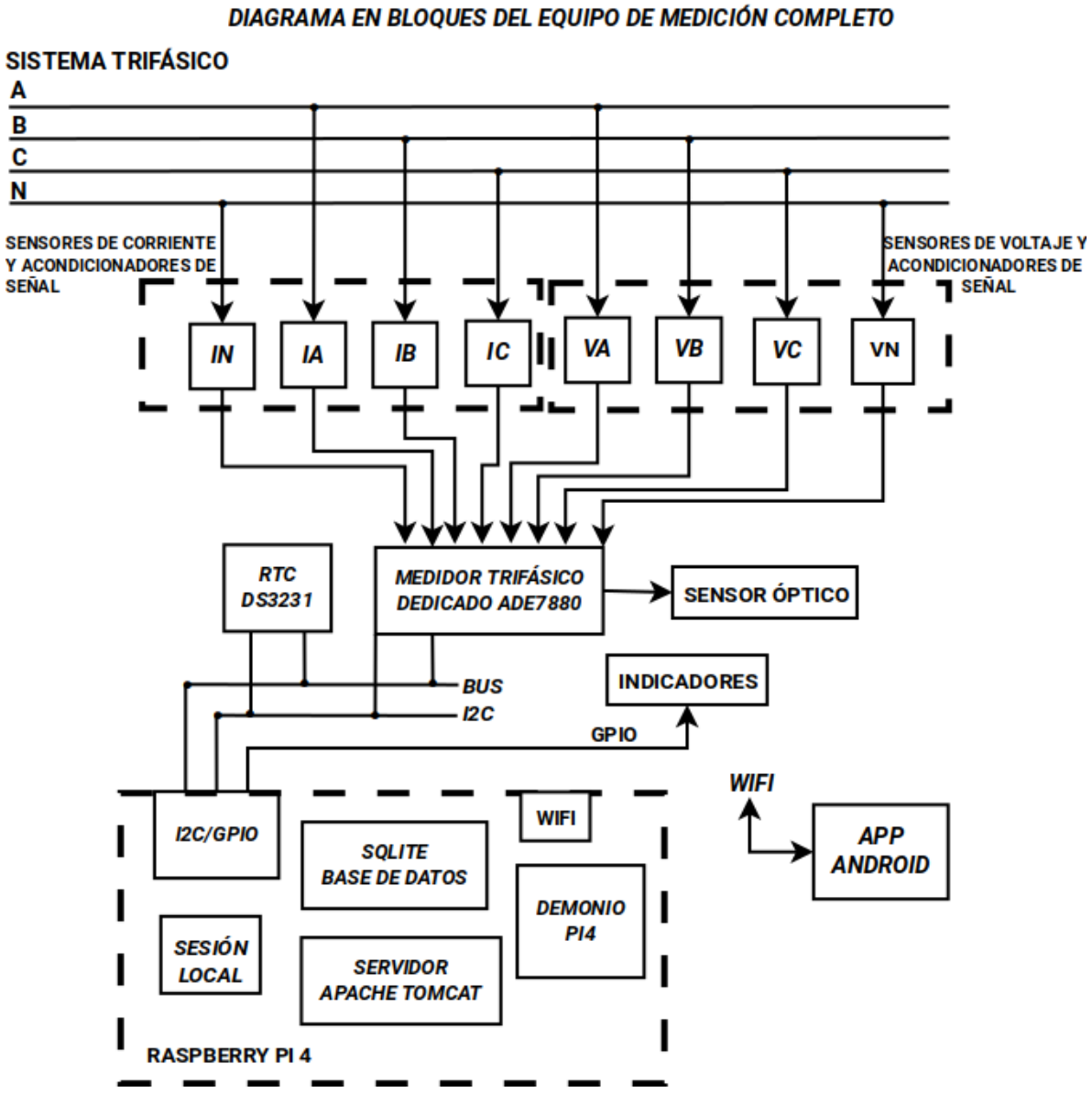


Figura 56: Diagrama en bloques del equipo de medición.

2.1 Sensado y acondicionador de señales de corriente

Como sensor de corriente se ha seleccionado el transformador SCT-013, su esquema eléctrico se muestra en la figura 58. Con una relación de transformación de 100A/50mA. Entrada de corriente/salida de corriente por lo que en el diseño del circuito se consideran las resistencias de burden para poder convertir la magnitud de corriente en el secundario de TC a un nivel equivalente de voltaje y este conectarlo a los pines de entrada de corriente del CI ADE7880. En la figura 57 se muestra el circuito que contienen los bloques IN, IA, IB e IC que conforman los sensores de corriente y los acondicionadores de señales de corriente. Con los valores de

las resistencias de burden se garantiza que el nivel de voltaje a plena escala sea el adecuado según las especificaciones técnicas ±0.5[Vp]. En el circuito se observa también un filtro pasivo paso bajo RC con una frecuencia de corte de 4.82[kHz]. Este filtro limita las señales de frecuencias superiores a la frecuencia de corte. También evita el fenómeno del aliasing durante el proceso de muestreo.

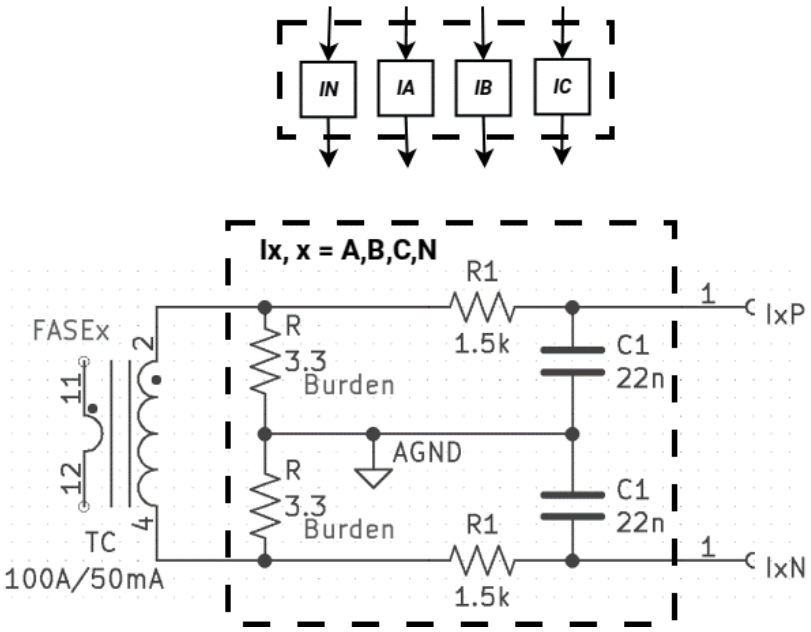


Figura 57: Circuito de adquisición para señales de corriente.

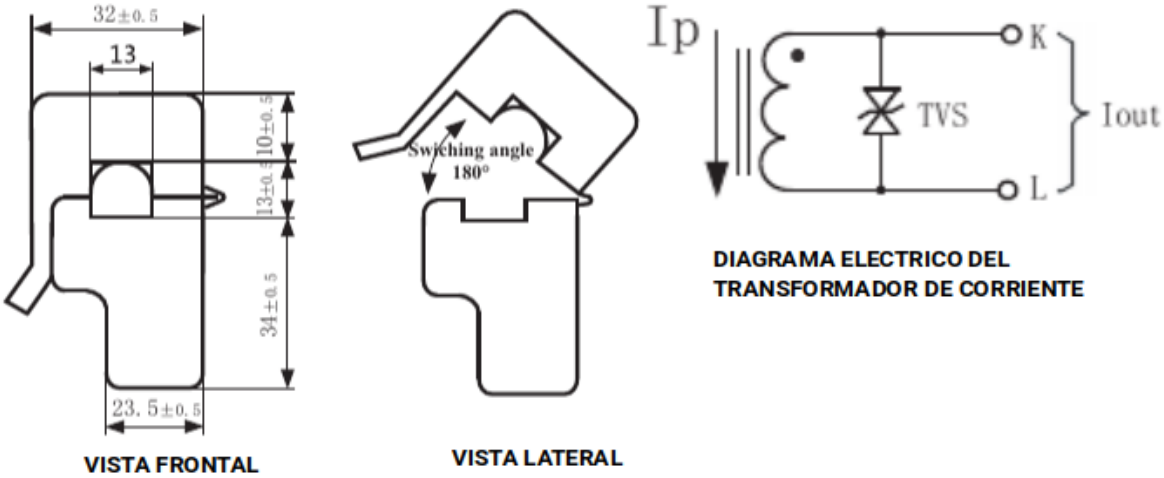


Figura 58: Diagrama mecánico y eléctrico del TC SCT-013.

2.2 Acondicionador de señales de voltaje

El circuito de voltaje está construido para proporcionar una señal de ±0.5[Vp] que cumpla con las especificaciones de entrada del ADE7880. Este circuito se conecta a los pines de entrada de voltaje del CI ADE7880, además se implementa un filtro pasivo paso bajo RC con una frecuencia de corte de 4.82[kHz]. El diseño del acondicionador de voltaje consiste de un arreglo resistivo configurado como un divisor de tensión, este divisor tiene la capacidad de operar en un rango de entrada de 120/240[Vrms]. Por lo que el equipo de medición está limitado a operar en esos niveles de tensión en sus entradas. En la figura 59 se muestra el circuito que contienen los bloques VA, VB, VC y VN cuando el sistema trifásico tiene una configuración en estrella. Según el diagrama en bloques general. Los valores de resistencias garantizan el nivel de voltaje a plena escala. Al igual que en el acondicionador de señal para la corriente. La función del filtro paso bajo es limitar las señales de voltaje de frecuencias por encima de la frecuencia de corte y evitar el fenómeno del aliasing en el muestreo.

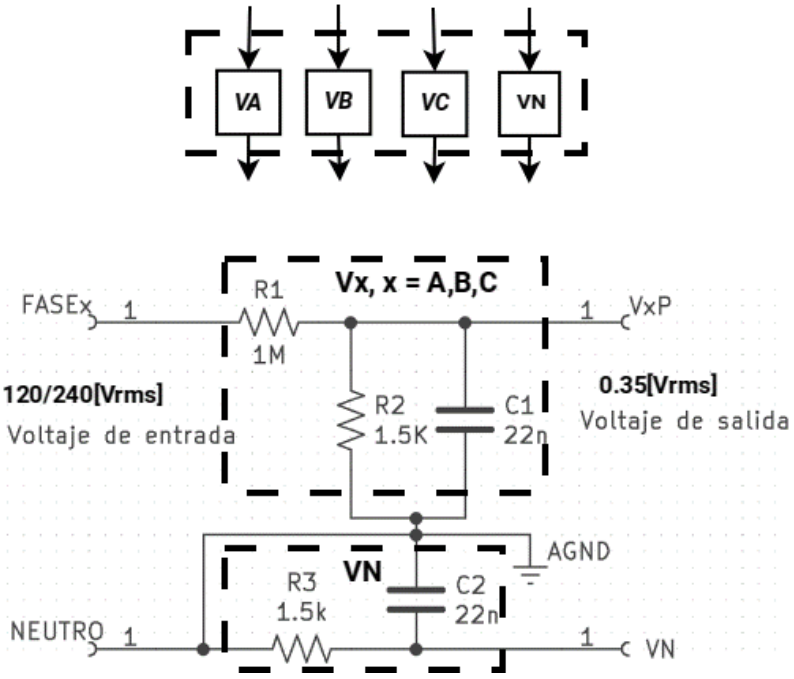


Figura 59: Circuito de adquisición para señales de voltaje.

2.3 Bus de comunicación I2C

Los dos CI que se conectan al puerto de comunicaciones I2C, son el reloj de tiempo real RTC DS3231 y el medidor trifásico ADE7880. La mini computadora raspberry pi cuenta con este puerto de comunicaciones. Véase la figura 47, los pines 3 y 5 forman el bus I2C según la distribución de pines del puerto GPIO. Para la conexión de estos dos dispositivos electrónicos se hace uso de cuatro líneas de este puerto. Dos lineas que forman el bus I2C y dos líneas de suministro de energía. Según la numeración de los pines: pin 1: suministro de voltaje

3.3[Vdc], pin 3: SDA del bus I2C, pin 5: SCL del bus I2C y un pin 9 de conexión a DGND. Como se muestra en la figura 60. También se muestra la salida por consola del comando i2cdetect el cual proporciona información acerca de las direcciones presentes en el bus I2C para cada dispositivo electrónico conectado. Por lo tanto al CI ADE7880 le corresponde la dirección en hexadecimal 0x38 y para el RTC DS3231 son las direcciones en hexadecimal 0x57 y 0x68. En este caso particular el RTC ya está configurado como reloj del sistema por lo que aparece UU sustituyendo la dirección 0x68 lo que significa que el kernel está haciendo uso del CI. En el bus de comunicaciones I2C la raspberry pi es el maestro siendo el RTC y el ADE7880 los esclavos, según la jerarquía al establecer comunicación.

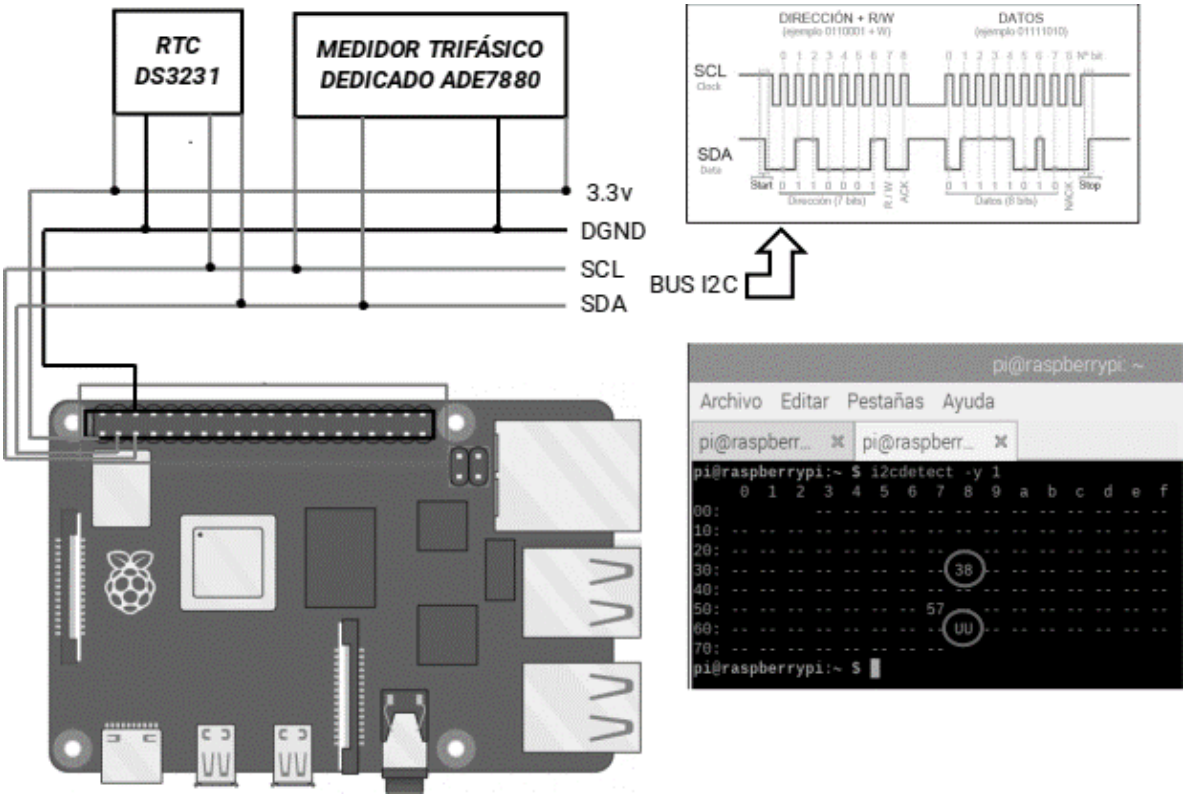


Figura 60: Conexiones al bus I2C.

Para poder realizar las operaciones de lectura y escritura en el CI ADE7880 se necesita seleccionar el puerto I2C, esto se logra escribiendo en el registro CONFIG2 el siguiente valor en hexadecimal 0x02, con esto se asegura de establecer en el bit 1 I2C_LOCK un uno lógico. Una vez habilitado el puerto de comunicaciones I2C, se pueden realizar las operaciones de lectura y escritura. En la figura 61 se muestra el diagrama de flujo para la operación de escritura. Se inicia con la comprobación del puerto de comunicaciones I2C, para esta operación es necesario haber habilitado el puerto I2C desde la configuración del sistema

operativo raspbian(para más detalles de la instalación del S.O. ver el anexo A), luego de haber hecho los ajustes correspondientes se tiene que reiniciar la raspberry pi. Como siguiente paso se escribe en el bus I2C la dirección del dispositivo con el cual se quiere establecer comunicación en este caso es con el medidor dedicado ADE7880 y su dirección es en hexadecimal 0x38, a continuación se escribe en el bus I2C la dirección del registro interno de interés y seguidamente los datos a escribir. Esta parte se realiza declarando un objeto del tipo I2CBUFFER que puede ser de 1, 2, 3 o 4 bytes según el registro de interés y finalmente se cierra el puerto I2C.

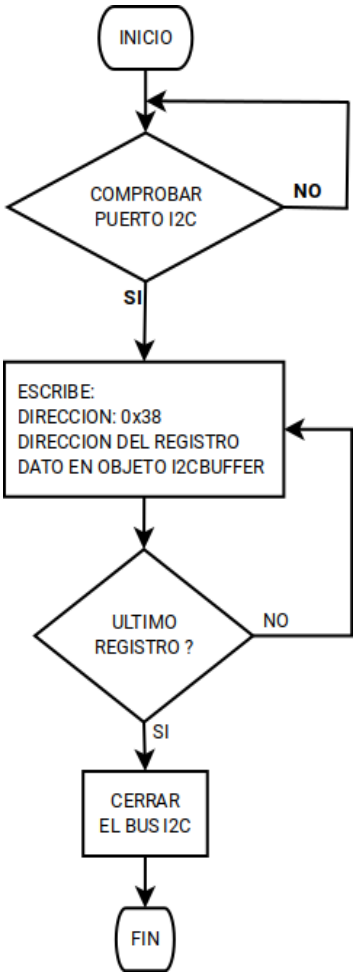


Figura 61: Operación de escritura en el ADE7880.

En la figura 62 se muestra el diagrama de flujo para la operación de lectura. Se hace un ejercicio de lectura completo desde la obtención de los datos del ADE7880 hasta su almacenamiento en la base de datos correspondiente.

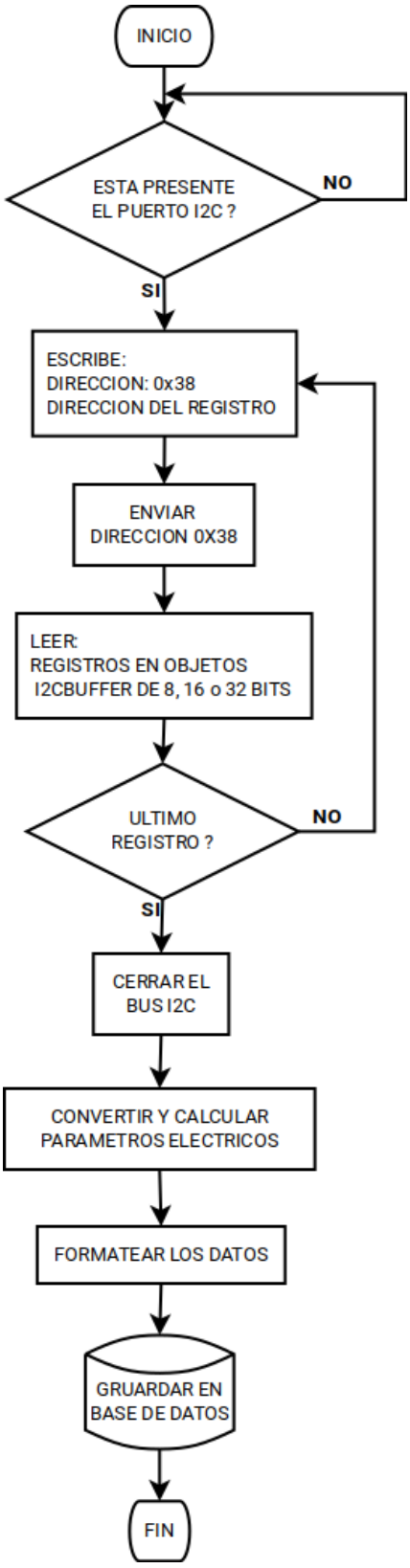


Figura 62: Operación de lectura ADE7880.

De la figura 62 se inicia con la comprobación del puerto de comunicaciones I2C. Como siguiente paso se escribe en bus I2C la dirección del dispositivo con el cual es requerido establecer comunicación en este caso es con el medidor dedicado ADE7880 y su dirección es en hexadecimal 0x38, a continuación se escribe en el bus I2C la dirección del registro interno de interés y seguidamente se escribe nuevamente en el bus I2C la dirección 0x38, luego se leen los registros internos. Esta parte se realiza declarando un objeto de tipo I2CBUFFER que puede ser de 1, 2 o 4 bytes según el registro de interés luego se cierra el puerto I2C. En este punto se deben convertir los datos de un objeto tipo I2CBUFFER a números enteros según sea el registro considerado. Para esta conversión se han creado dos métodos el primero es convertirByte32aINT y el segundo es convertirByte16aINT. Estos métodos reciben un objeto de tipo I2CBUFFER y retornan un numero entero en, a continuación se calculan los valores de los parámetros eléctricos, luego se les da formato a las mediciones calculadas quedando con dos cifras decimales. En este punto las mediciones están listas para ser almacenadas en la base de datos correspondiente, para esto se utiliza el entorno de persistencia creado para acceder a la base de datos y finalmente se cierra el ciclo de lectura y almacenamiento de mediciones.

2.3.1 Habilitar el puerto I2C en la raspberry pi 4

Para habilitar el puerto I2C se hace desde una terminal en raspbian, se escribe en la terminal el comando raspi-config el cual muestra una ventana con diferentes opciones del sistema tal como se muestra en la figura 63, se selecciona la opción 3 interface options, la cual carga un menú donde se muestran todas las interfaces soportadas por raspberry pi 4 según muestra la figura 64. En dicho menú se selecciona la interface I2C y se presiona la tecla enter se pide confirmación si se desea habilitar el puerto I2C como se muestra en la figura 65 tras una respuesta afirmativa se debe reiniciar, luego de esto el soporte queda habilitado permanentemente.

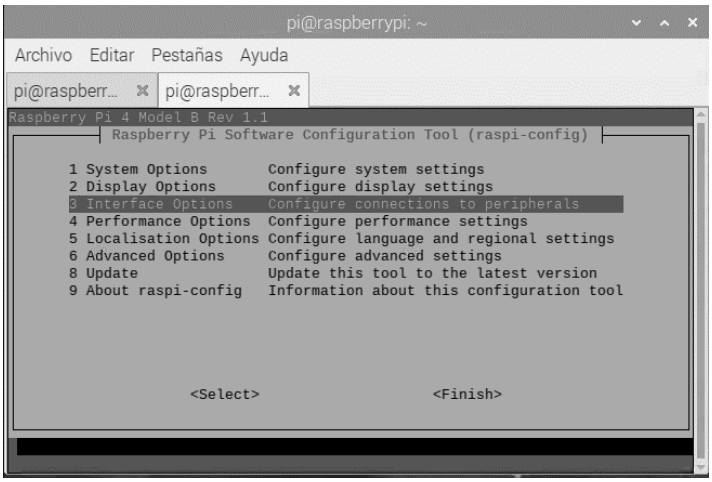


Figura 63: Habilitar puerto I2C opción 3.

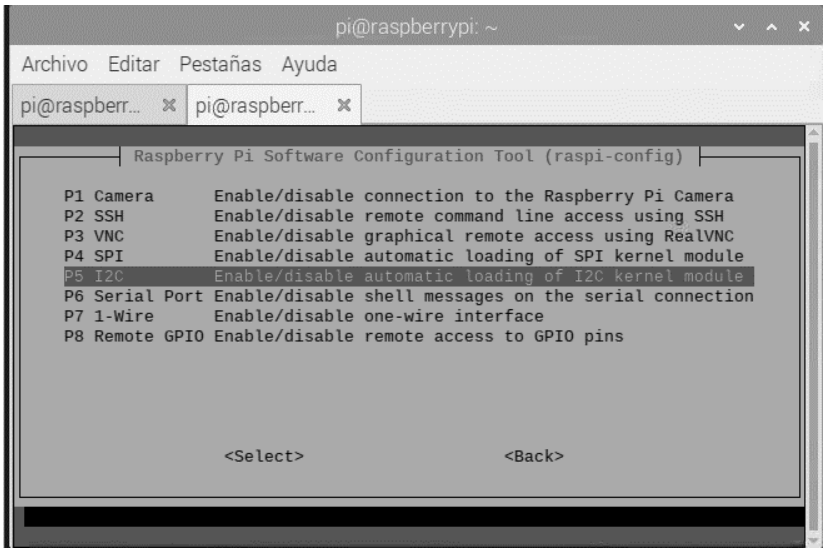


Figura 64: Habilitar puerto I2C opción P5.

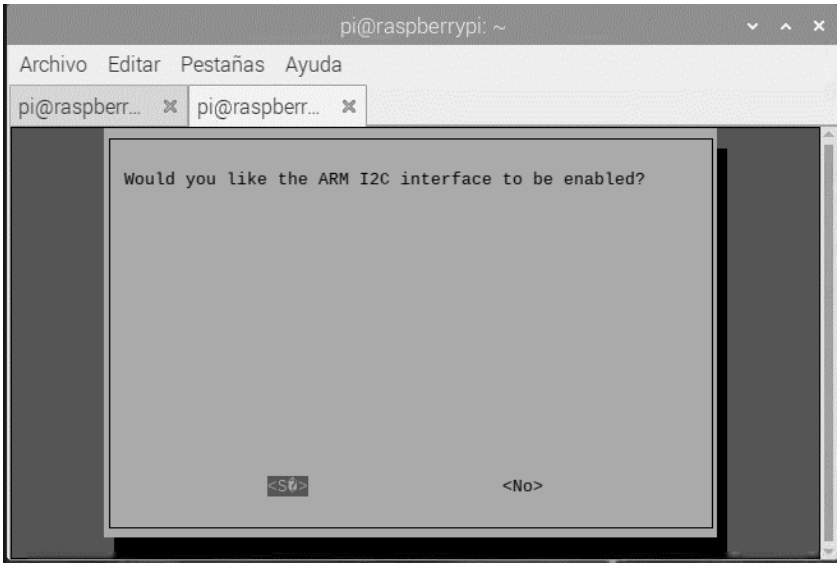


Figura 65: Habilitar puerto I2C.

2.3.2 Configuración del RTC como reloj del sistema

Para realizar esta configuración es necesario instalar el modulo del kernel rtc_ds1307 que le da soporte a varios tipos de RTC en un sistema operativo GNU/LINUX. En este caso particular el RTC que se utiliza es el DS3231, una parte de los requisitios para configurar el RTC como reloj del sistema es la instalación del módulo y la otra parte es editar algunos archivos del sistema. El RTC-DS3231 es un reloj calendario en tiempo real con comunicación I2C y de ser necesario más información remítase al anexo B. Para dar

comienzo se necesita conectar el RTC-DS3231 como se muestra en la figura 60, luego se debe instalar el paquete de software i2c-tools desde el instalador de paquetes del sistema. Se abre un terminal y se ejecuta el siguiente comando sudo mousepad /etc/modules y se abrirá un archivo el cual debe ser editado, la edición queda como se muestra en la figura 66. En versiones recientes del kernel se debe editar el archivo /boot/config.txt y añadir las dos líneas siguientes dtparam=i2c1=on y dtparam=i2c_arm=on.

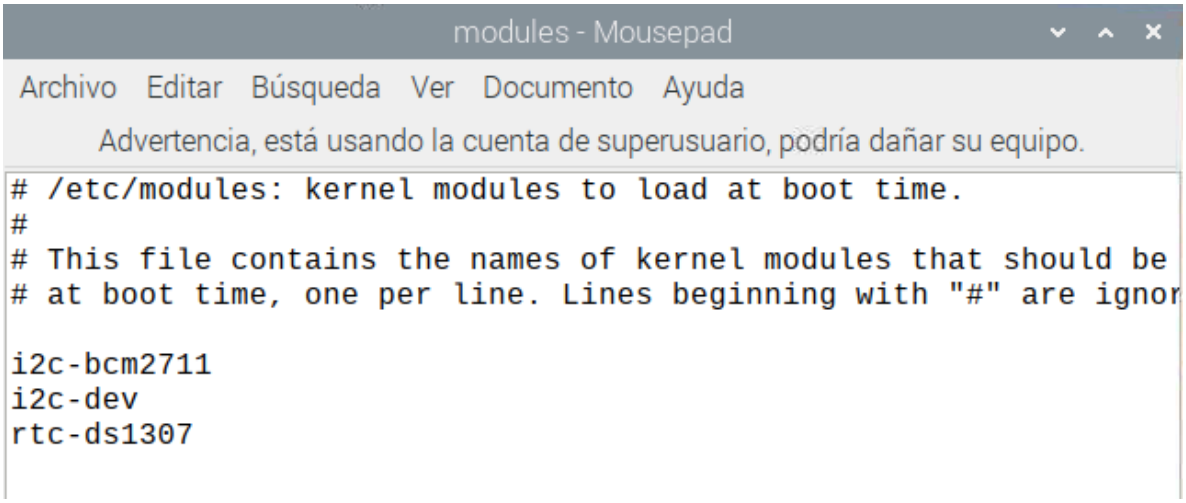


Figura 66: Cargando los módulos del kernel necesarios para el RTC-DS3231.

Ahora se debe reiniciar la raspberry pi 4 para que se carguen los módulos del kernel al S.O. Para tener seguridad que se configure y se establezca la hora correcta se edita el fichero /etc/rc.local y se añade lo siguiente como se muestra en la figura 67

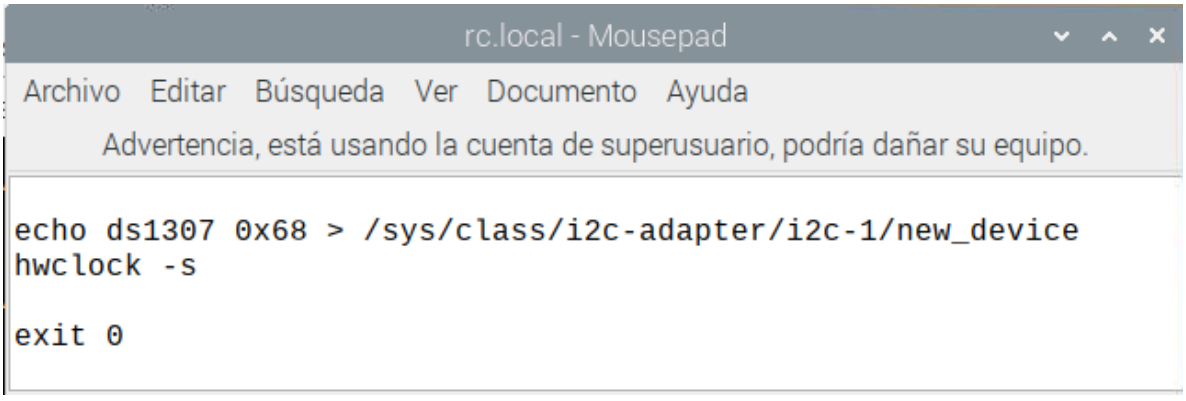


Figura 67: Configuración del RTC-DS3231 editando el fichero rc.local.

.

Finalmente se comprueba que el módulo RTC-DS3231 está cargado correctamente en el S.O. Como se muestra en la figura 68.

```
pi@raspberrypi:~ $ dmesg | grep rtc
[    7.785830] rtc-ds1307 1-0068: registered as rtc0
pi@raspberrypi:~ $
```

Figura 68: Configuración terminada del RTC-DS3231.

2.4 Conexión del sensor óptico

Los dispositivos electrónicos que forman el sensor óptico se muestran en la figura 69, estos son la resistencia R y el diodo emisor de luz D, la resistencia tiene un valor de 220[Ω] el led seleccionado emite luz roja en la banda de los 660[nm]. Como se muestra en el diagrama en bloques el sensor óptico depende exclusivamente del CI ADE7880, en los pines de salidas CFx. El pin 33 corresponde a la salida CF1 y es el que se utiliza para proporcionar lecturas de energía. El circuito que conforma el sensor óptico se conecta a una fuente de voltaje de corriente directa de 3.3[Vdc]. El sensor óptico emite pulsos de duración variable que proporciona información de la energía acumulada. La duración de los pulsos está determinada por la configuración hecha en los registros CFx véase el apartado 1.9.1.17 Conversión de energía a frecuencia para más detalles.

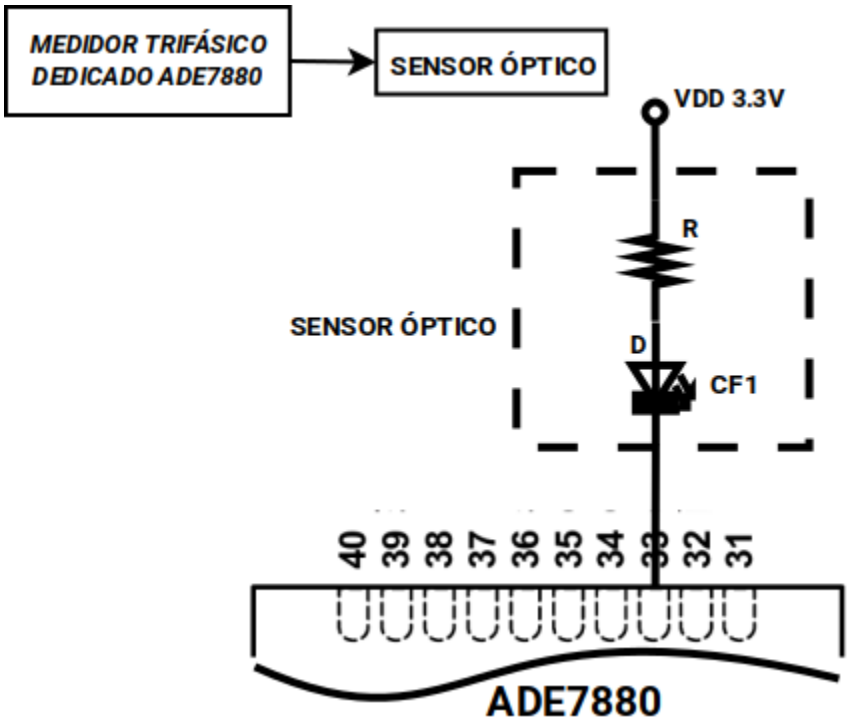


Figura 69: Conexionado del sensor óptico.

2.5 Indicadores del equipo de medición

Los indicadores presentes en el equipo de medición son los encargados de proporcionar el estado de operación y configuración que se le ha hecho al medidor. Los indicadores con los que este diseño cuenta son cuatro estos indican los tipos de conexión trifásica soportados que pueden ser conexiones en delta o en estrella. También proporcionan información acerca del encendido y apagado del medidor, se cuenta con otro indicador que genera información de la habilitación de las conexiones remotas.

- Los indicadores del tipo de conexión trifásica son dos diodos led, que dependiendo de cuál conexión se configure así enciende el indicador correspondiente.
- El indicador de encendido y apagado entra en funcionamiento una vez haya entrado en ejecución el programa principal.
- El indicador que habilita las conexiones remotas se activa en las opciones de configuración del equipo de medición trifásico.

En la figura 70 se muestra el diagrama de conexiones de los indicadores con los pines del puerto GPIO. Para más detalles de este puerto consultar la sección 1.10.1 Puerto de entrada y salida de propósito general. Los pines utilizados para controlar los indicadores se han configurado como salidas digitales.

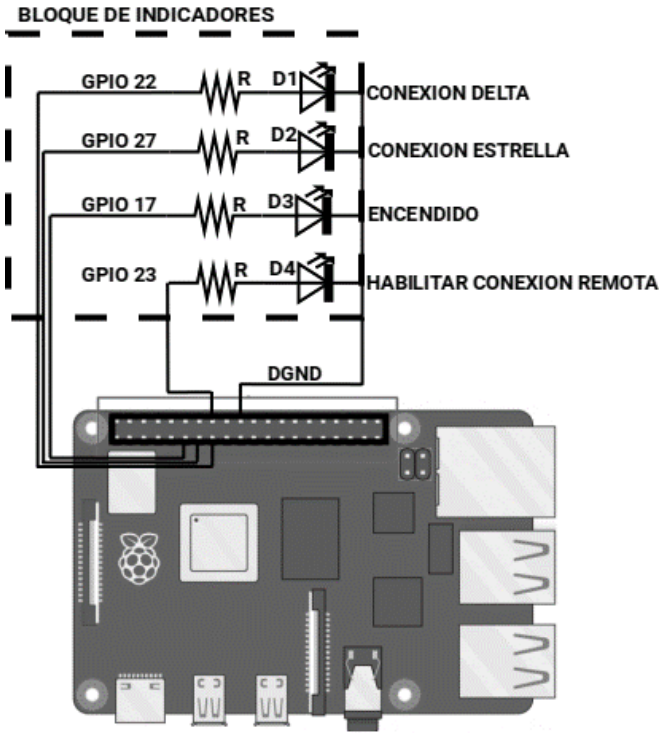


Figura 70: Conexiones de los indicadores led con el puerto GPIO.

2.6 Almacenamiento físico de las mediciones

El dispositivo en el que se almacenan las mediciones es una memoria micro sd de 32GB, esta memoria se instala en una ranura especial que trae en la parte inferior la placa raspberry pi como se muestra en la figura 71, para más detalles revisar la sección 1.10 Raspberry pi ordenador de placa única, esta memoria es el disco duro de la mini computadora raspberry pi en la cual se instala un sistema operativo, para este caso en particular el S.O. es raspbian. Llegados a este punto ya se cuenta con el soporte de un computador convencional, en cuanto al sistema de archivos, instalación de software como por ejemplo bases de datos. En este trabajo se utiliza el gestor de bases de datos sqlite y es por medio del cual que se almacenan las mediciones. Esta etapa se explica más en detalle en el capítulo tres.

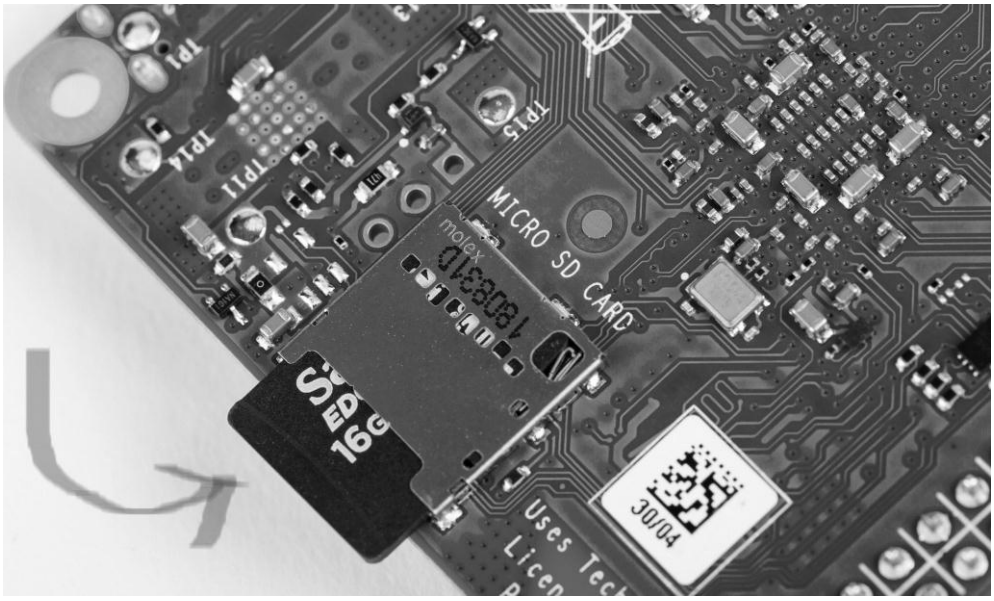


Figura 71: Ranura para memorias micro sd, en RPI 4.

2.7 Construcción de la carcasa del equipo de medición

La elaboración de la carcasa donde se montan los circuitos que conforman el equipo de medición ha sido fabricada en una impresora 3D, se han utilizado dos tipos de filamentos el primero llamado PLA o ácido poliláctico. Para construir la mayor parte de la carcasa de color naranja. El segundo tipo de filamento es el ABS o acrilonitrilo butadieno estireno. Con el que se ha fabricado la tapa del equipo de medición en color negro. Las propiedades más importantes de ambos filamentos se muestran en la tabla 7.

PROPIEDAD	PLA	ABS
Facilidad de impresión	Muy alta	alta
Temperatura de extrusión	(180 – 230)°C	(210 – 250)°C
Temperatura de la cama	(20 – 60)°C, opcional	(80 – 110)°C
Resistencia a la tracción	58MPa	(35 – 50)MPa
Resistencia a la flexión	55MPa	(50 – 87)MPa
Densidad	(1.23 – 1.25)g/cm^3	(1.02 – 1.08)g/cm^3
Temperatura de deformación	Baja, 55°C	Media, 80°C

Tabla 7: Propiedades más importantes de los filamentos para la impresión 3D.

En la figura 72 se muestran las dimensiones de la carcasa ya impresa.

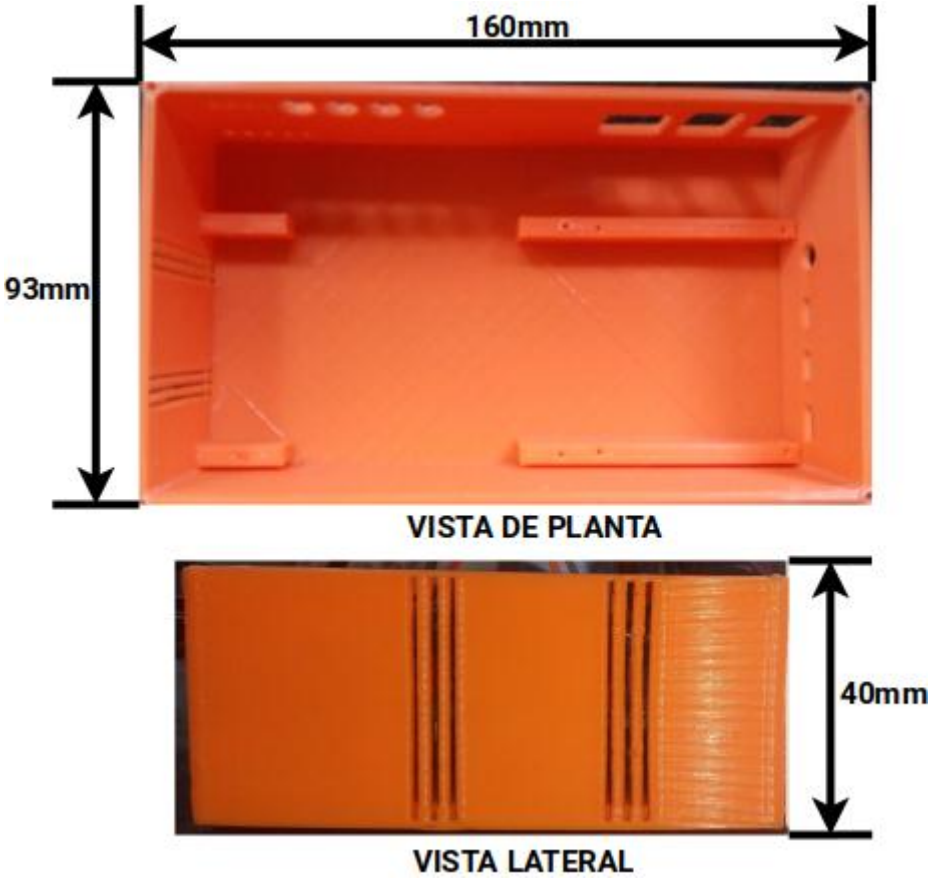


Figura 72: Dimensiones de la carcasa impresa en 3D.

En la figura 73 se muestra la distribución de entradas de la carcasa para el equipo de medición.

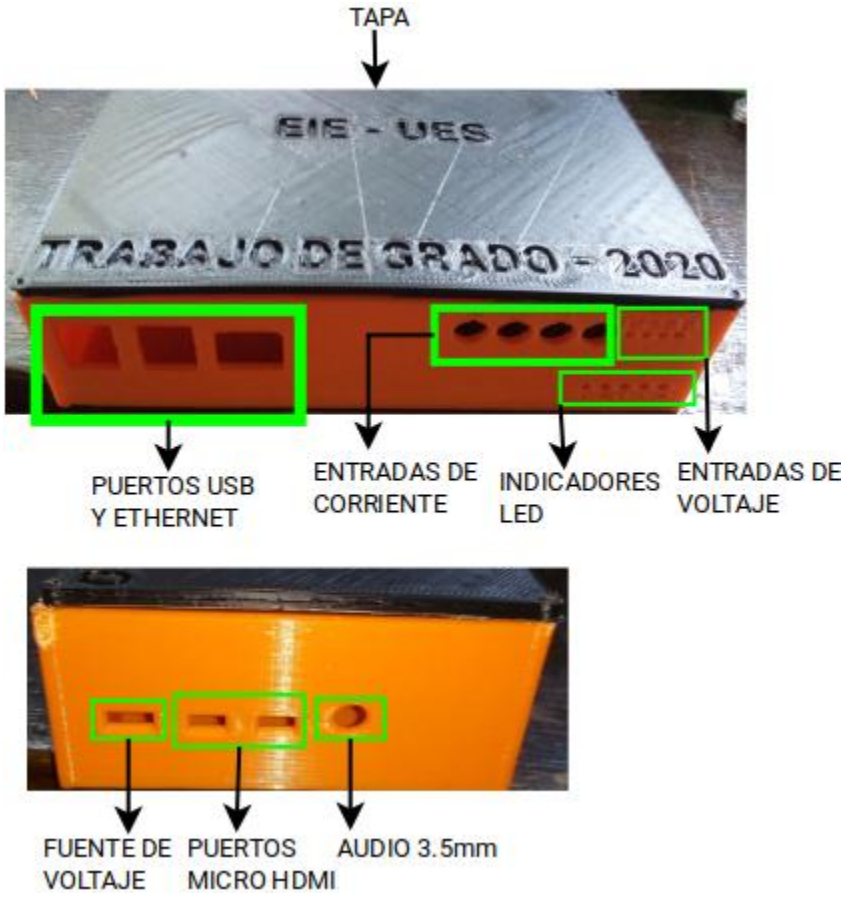


Figura 73: Distribución de las entradas en la carcasa.

En la figura 74 se muestran los circuitos montados en la carcasa.

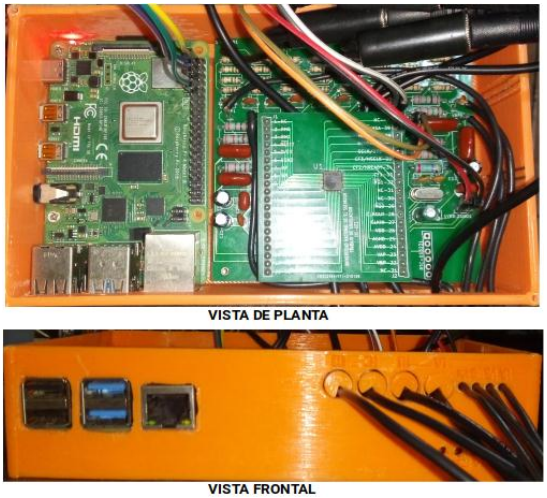


Figura 74: Montaje de circuitos del equipo de medición.

2.8 Fuente de alimentación

El suministro de energía para la mini computadora raspberry pi se hace a través de una fuente de corriente directa de cuarta generación de raspberry pi tipo C, en la figura 75 se muestran sus dimensiones. Esta fuente opera en un rango de voltaje de entrada que va 100 – 240V y a frecuencias de 50/60Hz. Proporcionando a su salida 5Vdc y una corriente máxima de 3A. Además cuenta con las siguientes características de protección contra:

- Sobre carga.
- Sobre corriente.
- Cortocircuitos.
- Sobre tensión.
- Bajo voltaje.
- Electrostática.

El modelo de la fuente oficial para las mini computadoras raspberry pi es JY15-050-300-UD. La alimentación del medidor ADE 7880 se realiza por medio del puerto GPIO desde el cual se tienen disponibles niveles de voltaje de 3.3Vdc.

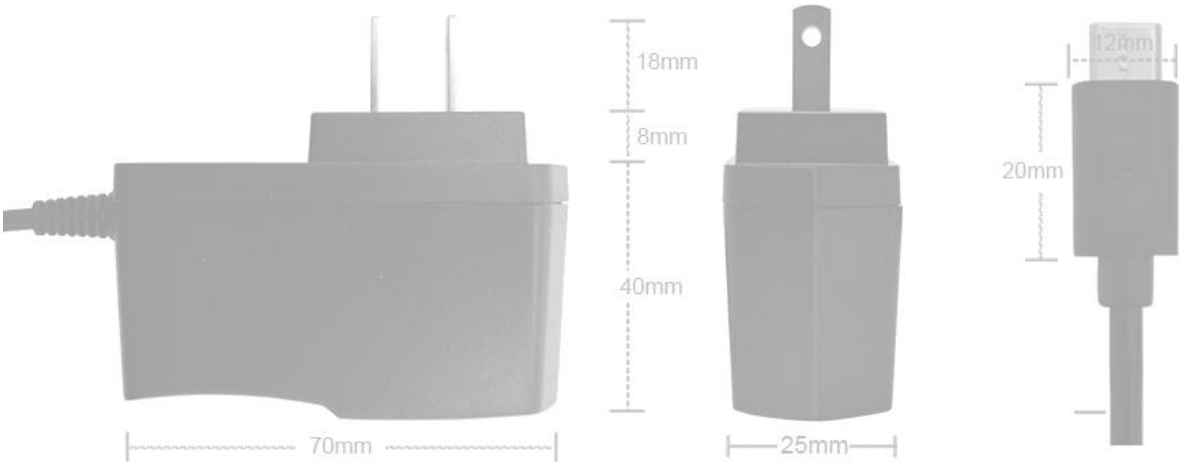


Figura 75: Dimensiones de la fuente de alimentación JY15-050-300-UD.

CAPÍTULO III

DISEÑO Y DESARROLLO DEL SOFTWARE PARA EL HARDWARE PROTOTIPO

3.0 Introducción

En este capítulo se describe en detalle el diseño y desarrollo de la solución por software para dar soporte a la extracción de mediciones desde el CI ADE7880 vía el puerto de comunicaciones I2C así como la conversión y cálculo de los valores medidos, además de formatear los datos antes de ser almacenados en la base de datos y el desarrollo de un sistema informático para la autenticación de sesiones. La interacción con la base de datos se realiza utilizando persistencia. Se hace uso de indicadores tipo led para obtener información acerca del estado del medidor trifásico. Como servidor de aplicaciones se utiliza apache tomcat, utilizado como soporte que permite establecer comunicación con la aplicación móvil desarrollada, para equipos móviles Android. Esta aplicación tiene la función de mostrar una tabla con todas las mediciones almacenadas en la base de datos y además mostrar graficas de las mediciones. En la figura 76 se muestran dos tipos de diagramas A y B que resumen la implementación por software para el equipo de medición trifásico.

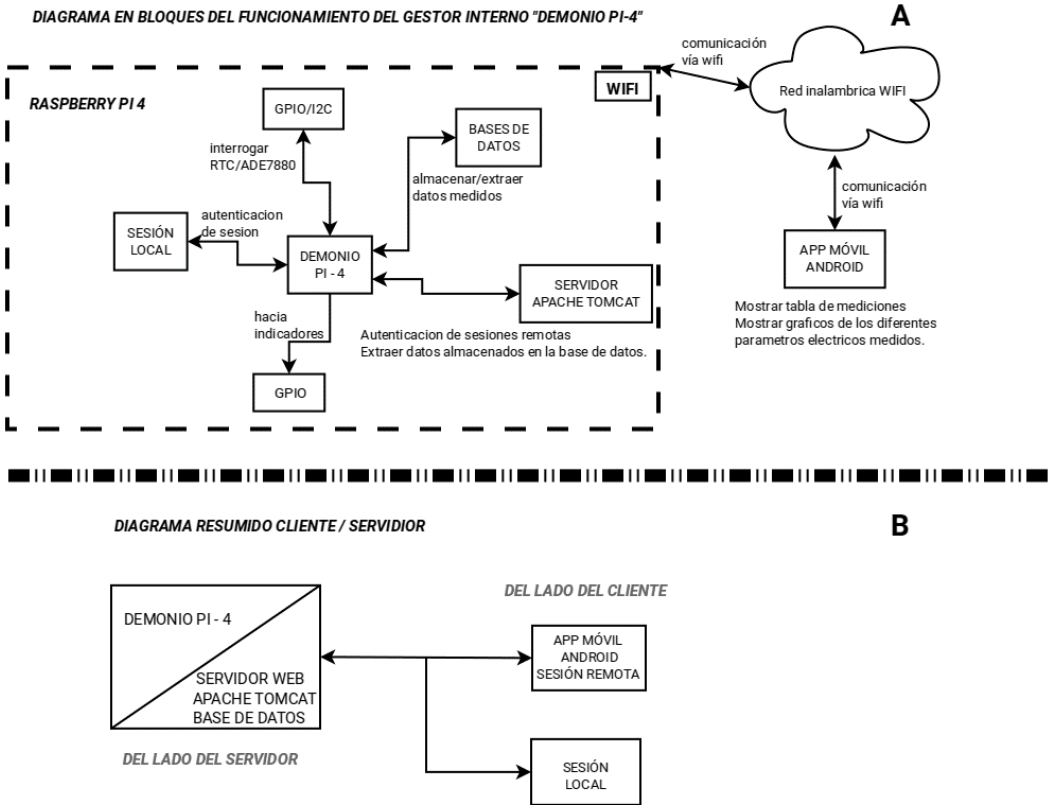


Figura 76: Diagrama en bloques de la solución por software.

La parte A del diagrama de bloques muestra el funcionamiento del gestor interno al que se le ha llamado demonio pi-4. Este muestra una visión general de las partes más importantes del desarrollo de software necesario. En la parte B se muestra un diagrama cliente servidor. El

desarrollo de software es 100% con tecnología java tanto para la aplicación embebida en la placa raspberry así como en la aplicación del cliente móvil.

3.1 Implementación de la sesión local (stand-alone)

La sesión local se desarrolla para darle al equipo de medición cierto grado de autonomía y privacidad, debido a la importancia de la información que se maneja. Entre los paquetes necesarios para el desarrollo de la sesión local se encuentran: java.utils y javax.swing. Utilizados entre otras tareas para crear listas interfaces grafica de usuario como ventanas y menús de opciones. Los usuarios que pueden acceder a la aplicación del equipo de medición deben haber sido definidos en la base de datos, estos usuarios pueden ser de dos tipos:

- Usuarios solo para consulta.
- Usuarios con privilegios o administradores del sistema.

Las opciones para crear y eliminar usuarios del sistema se encuentran disponibles, no se limita el número de usuarios registrados. La base de datos se integra haciendo uso de un motor de persistencia, esta tecnología nos permite interactuar con la base de datos sin la necesidad de escribir código SQL directamente. En su lugar se crean clases entidades para cada tabla de datos y de esta forma es posible interactuar con sus correspondientes registros como si fueran objetos de una clase. Este hace uso de las características de la POO aplicadas a las bases de datos relacionales permitiendo que las clases y los objetos de una aplicación puedan ser almacenados, modificados y buscados de forma eficiente en unidades de persistencia.

3.1.1 Configuración del entorno de persistencia

Como primer paso se selecciona un motor de persistencia que para este trabajo es EclipseLink, este motor de persistencia permite la creación de las clases de entidad y control que son vinculadas desde la base de datos existente. Luego de haber seleccionado el motor de persistencia este se debe incluir en las dependencias del proyecto. Para ello se despliegan todas las carpetas del proyecto y se selecciona la carpeta llamada dependencies se hace clic derecho para que muestre el menú contextual en el cual se selecciona la opción “add dependency” y esto muestra una nueva ventana en la cual se busca EclipseLink y se presenta una lista de proveedores desde la cual se selecciona el correspondiente. En la figura 77 se muestra la ventana para incluir en el proyecto las dependencias. Siguiendo el mismo proceso se instala el paquete sqlite-jdbc.

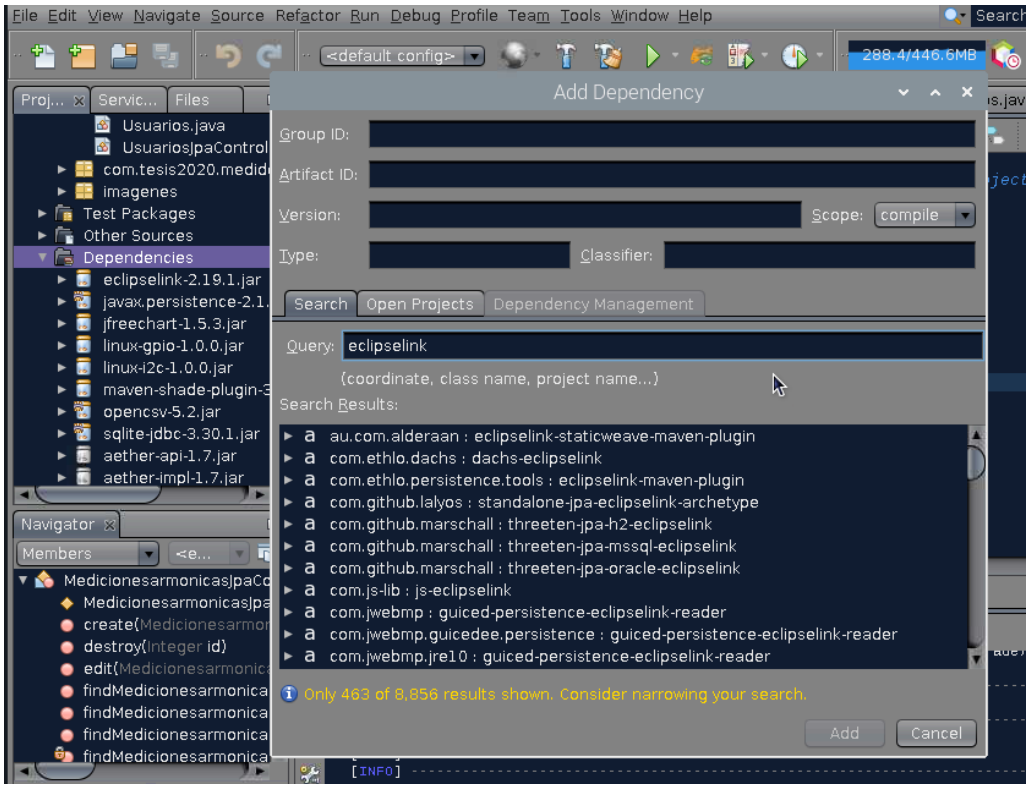


Figura 77: Incorporando la dependencia EclipseLink al proyecto.

Luego se deben crear las unidades de persistencia las cuales se definen en un archivo XML. En las carpetas del proyecto se selecciona "Other Sources/src/main/resources/META-INF/persistence.xml". En dicho archivo se definen los siguientes parámetros de importancia:

- El nombre de la unidad de persistencia.
- La implementación de JPA definido.
- La conexión a la base de datos.

En la figura 78 se muestra una parte del archivo persistence.xml

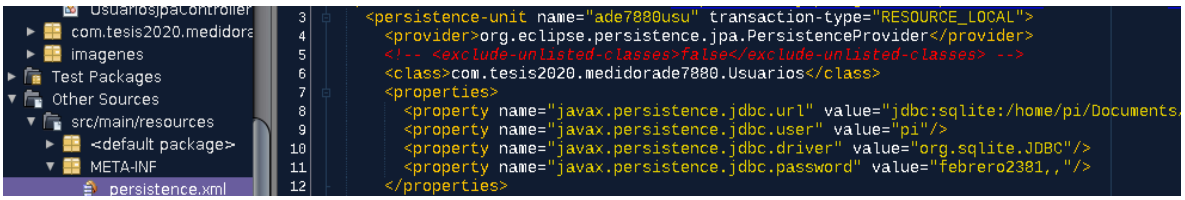


Figura 78: Archivo de configuración de unidades de persistencia.

En la línea 3 del archivo, mostrada en la figura 78 se asigna el nombre a la unidad de persistencia. En la línea 4 se define la implementación de JPA que se ha seleccionado en este caso EclipseLink. Entre las líneas 7 a 12 se muestran las propiedades para establecer la conexión con la base de datos. En este punto falta agregar las bases de datos al proyecto "MedidorADE7880" para ello se necesita el controlador sqlite-jdbc. Se selecciona la pestaña de servicios en la cual se muestra un menú en el que se selecciona la opción databases se hace clic derecho y se selecciona la opción "New conection" a continuación se carga una ventana donde es posible seleccionar o adicionar el driver sqlite-jdbc se presiona el botón next y se muestra otra ventana donde se pide ingresar el username, password y la dirección local donde se tenga la base de datos, para finalizar se presiona el botón finish. Con este proceso se ha adicionado la base de datos que se necesitan propias del proyecto. Como se muestra en la figura 79.

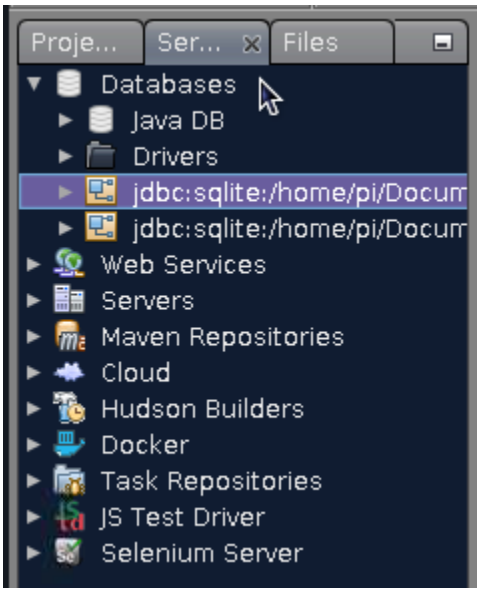


Figura 79: Adicionando las bases de datos.

3.1.2 Creación de las clases entidad y controladoras

Una de las ventajas que se tiene es utilizar JPA habilita la opción de interactuar con las bases de datos a través de objetos. Las entity class o clases entidad. Estas clases son creadas a partir del mapeo de una tabla en una base de datos, el mapeo se desarrolla mediante anotaciones. Estas anotaciones proporcionan los suficientes metadatos como para poder relacionar las clases con las tablas y las propiedades con las columnas. De esta forma es que se tiene la capacidad de interactuar con las bases de datos a través de clases. El proceso de creación de clases entidad que se ha utilizado en el presente trabajo es de forma automática usando el IDE de Netbeans. Para crearlas se debe posicionar sobre el paquete principal del proyecto que para este caso es com.tesis2020.medidorade7880 se hace clic derecho para acceder a un

menú en el cual se selecciona la opción New esta despliega un submenú del cual se selecciona la opción entity classes from database y se abre una ventana. Se seleccionan las bases de datos que previamente se han configurado en el proyecto. Por lo que se selecciona la base de datos SESIONES.BD por lo cual se muestran las tablas que están contenidas en ella. Como se muestra en la figura 80.

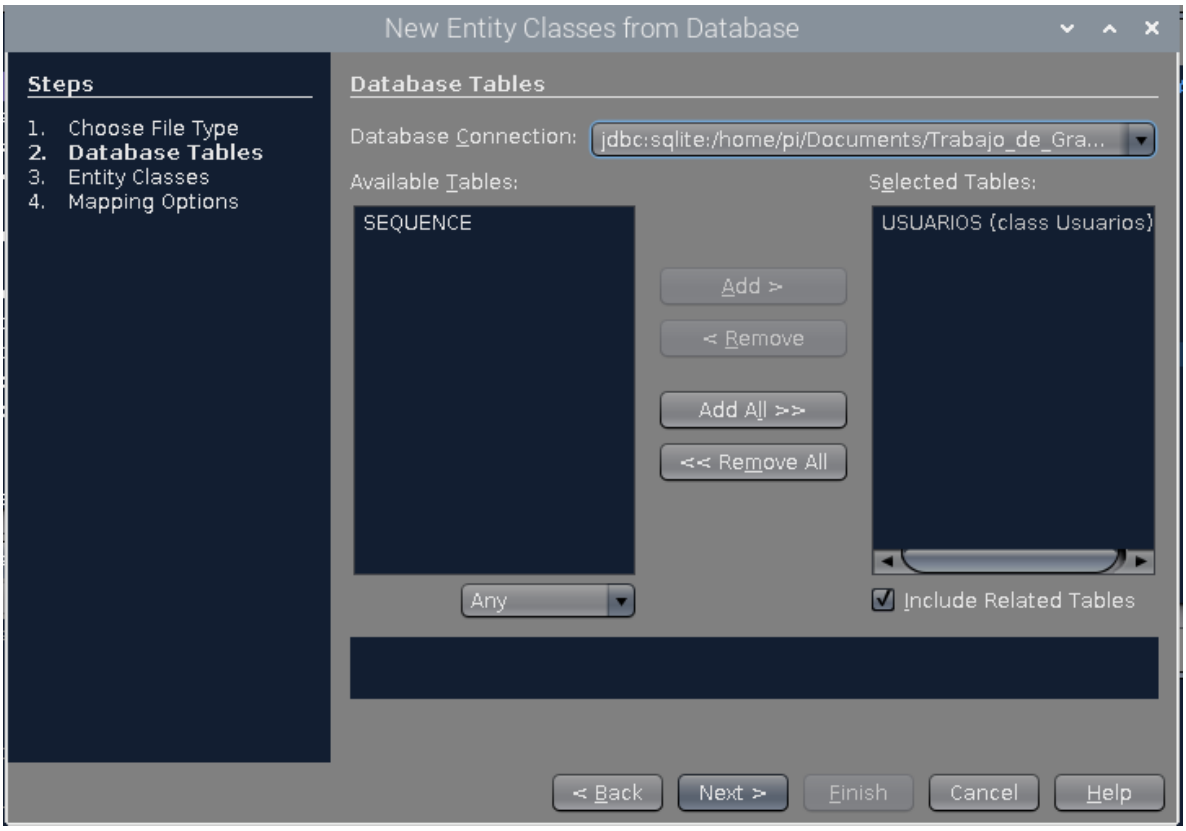


Figura 80: Creación de la clase entidad para la tabla USUARIOS.

La tabla de interés es USUARIOS la cual se selecciona y se presiona el botón next y en la siguiente ventana da la opción de elegir una ubicación donde guardar la clase, se presiona next y luego se presenta una ventana con opciones de mapeo dichas opciones se configuran como se muestra en la figura 81 y se presiona el boton finish. Con lo que se ha creado la clase entidad Usuarios.java. El contenido es el mapeo de la tabla USUARIOS con sus anotaciones respectivas, como se muestra en la figura 82.

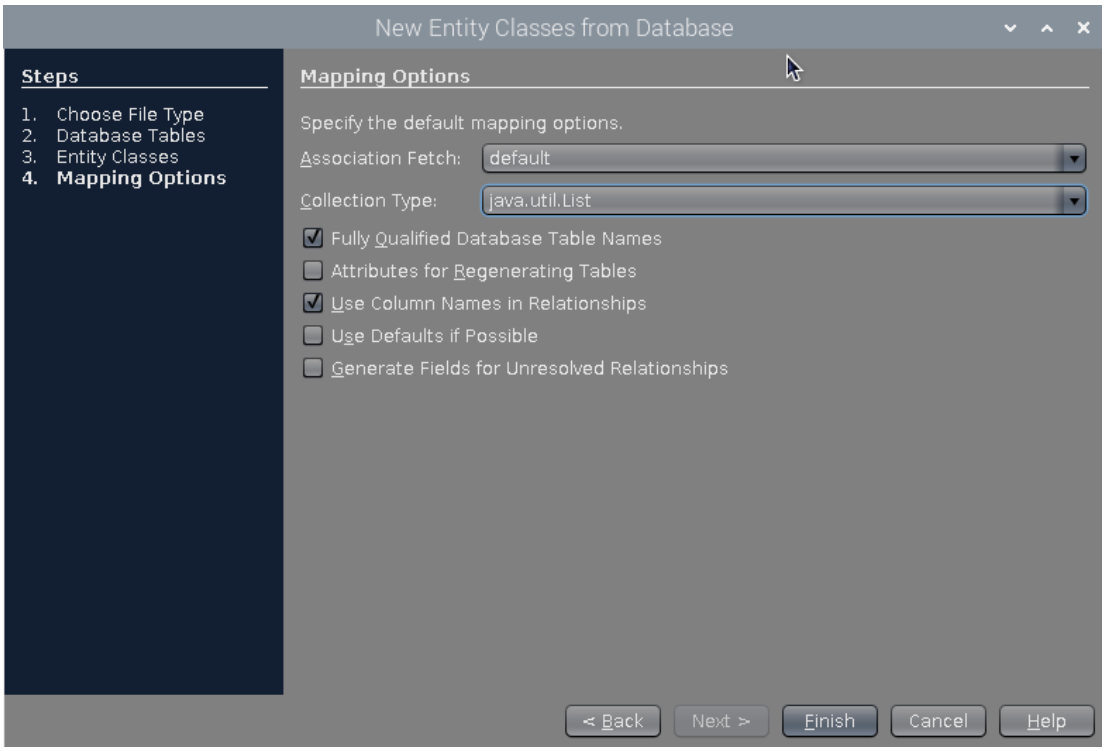


Figura 81: Opciones de mapeo para la clase entidad USUARIOS.

```
6   package com.tesis2020.medidorade7880;
7
8   import java.io.Serializable;
9   import javax.persistence.Basic;
10  import javax.persistence.Column;
11  import javax.persistence.Entity;
12  import javax.persistence.GeneratedValue;
13  import javax.persistence.GenerationType;
14  import javax.persistence.Id;
15  import javax.persistence.NamedQueries;
16  import javax.persistence.NamedQuery;
17  import javax.persistence.Table;
18
19  /**
20   *
21   * @author root
22   */
23  @Entity
24  @Table(name = "USUARIOS")
25  @NamedQueries({
26      @NamedQuery(name = "Usuarios.findAll", query = "SELECT u FROM
27      @NamedQuery(name = "Usuarios.findById", query = "SELECT u FRO
28      @NamedQuery(name = "Usuarios.findByEstampa", query = "SELECT
29      @NamedQuery(name = "Usuarios.findByUsuario", query = "SELECT
30      @NamedQuery(name = "Usuarios.findByClave", query = "SELECT u
31      @NamedQuery(name = "Usuarios.findByBandera", query = "SELECT
32  public class Usuarios implements Serializable {
```

Figura 82: Clase entidad USUARIOS.

En la línea 23 se observa la anotación @Entity la importancia de esta radica en que le indica a JPA que es una clase entidad y que por lo tanto debe ser administrada como tal. Es una forma para diferenciarse de las demás clases del proyecto. En la línea 26 se observan las anotaciones @NamedQuery que son consultas básicas. Una vez realizado lo anterior se procede a crear la clase usuariosjpacontroller.java que se encarga de manejar la clase entidad y es quien se conecta con las unidades de persistencia que se han definido en el archivo persistence.xml. Para crear la clase se debe posicionar sobre el paquete principal del proyecto que para este caso es com.tesis2020.medidorade7880 se da clic derecho para que despliegue un menú y se selecciona de este la opción New esta opción desplegara un submenú del cual se selecciona la opción JPA Controller Classes From Entity Classes y se desplegara una ventana donde se muestra un listado de las clases entidad, por lo que se selecciona la clase entidad de interés que en este caso es usuarios, luego se presiona el botón next y finalmente el botón finish y se ha creado la clase usuariosjpacontroller.java. Como se muestra en la figura 83.

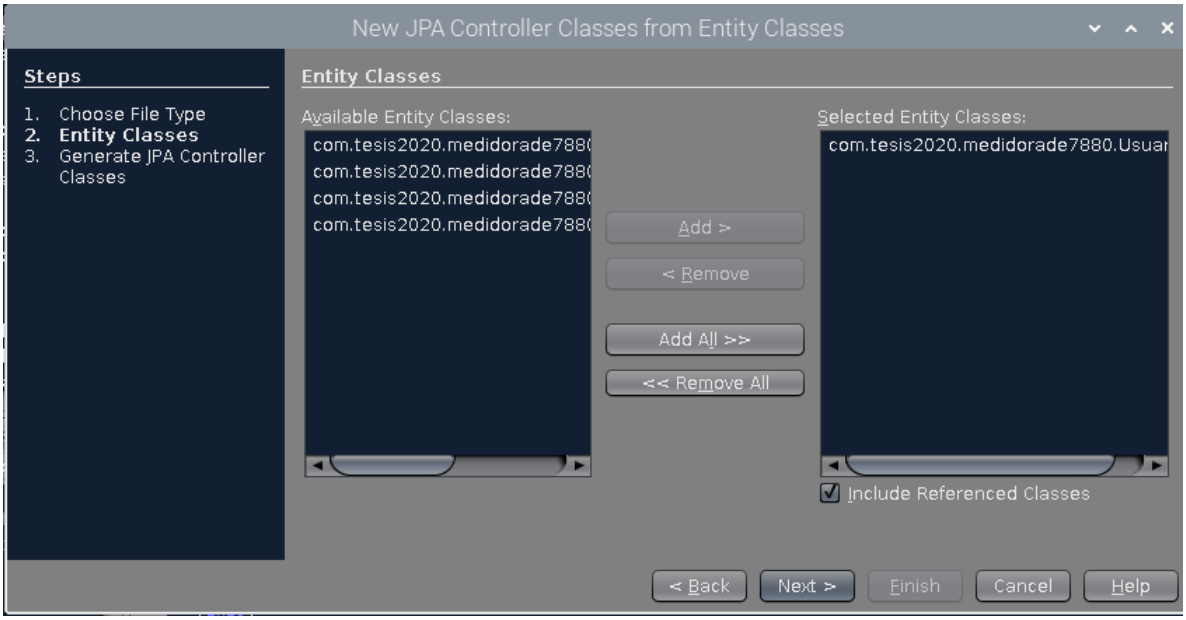


Figura 83: Creación de la clase ususariojpacontroller.

Esta clase se debe modificar para incluir un constructor que se conecte con una unidad de persistencia previamente definida. En la figura 84 se muestra como queda la declaración del constructor de la clase usuariojpacontroller.

```java
package com.tesis2020.medidorade7880;

import com.tesis2020.medidorade7880.exceptions.NonexistentEntityException;
import java.io.Serializable;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.Persistence;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

/**
 *
 * @author root
 */
public class UsuariosJpaController implements Serializable {

    public UsuariosJpaController() {
        this.emf = Persistence.createEntityManagerFactory("ade7880usu");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

```

Figura 84: Clase usuariosjpacontroller.

En las líneas 25 y 26 se crea el constructor de la clase y se crea una conexión con una unidad de persistencia la cual es llamada ade7880usu en el archivo persistence.xml.

3.1.3 Desarrollo de la interfaz gráfica de usuario

Para el desarrollo de la GUI se hace uso de las herramientas del paquete javax.swing que proporciona un conjunto de widgets para la creación de interfaces graficas de usuario. Los componentes gráficos que se usaran son:

- Jframe el cual será el encargado de contener los demás widgets.
- JtextField por medio del cual se introduce el nombre de usuario.
- JpasswordField por medio del cual se introduce la clave.
- Jlabels para poder introducir texto e imágenes.
- Jbutton para hacer la validación de la sesión.
- Separator horizontal.

Al unir los componentes gráficos y ordenarlos en el jframe la GUI queda como se muestra en la figura 85. La creación de GUIS en java con las herramientas

Graficas que proporciona el paquete swing hace de la implementación de interfaces una experiencia muy práctica.

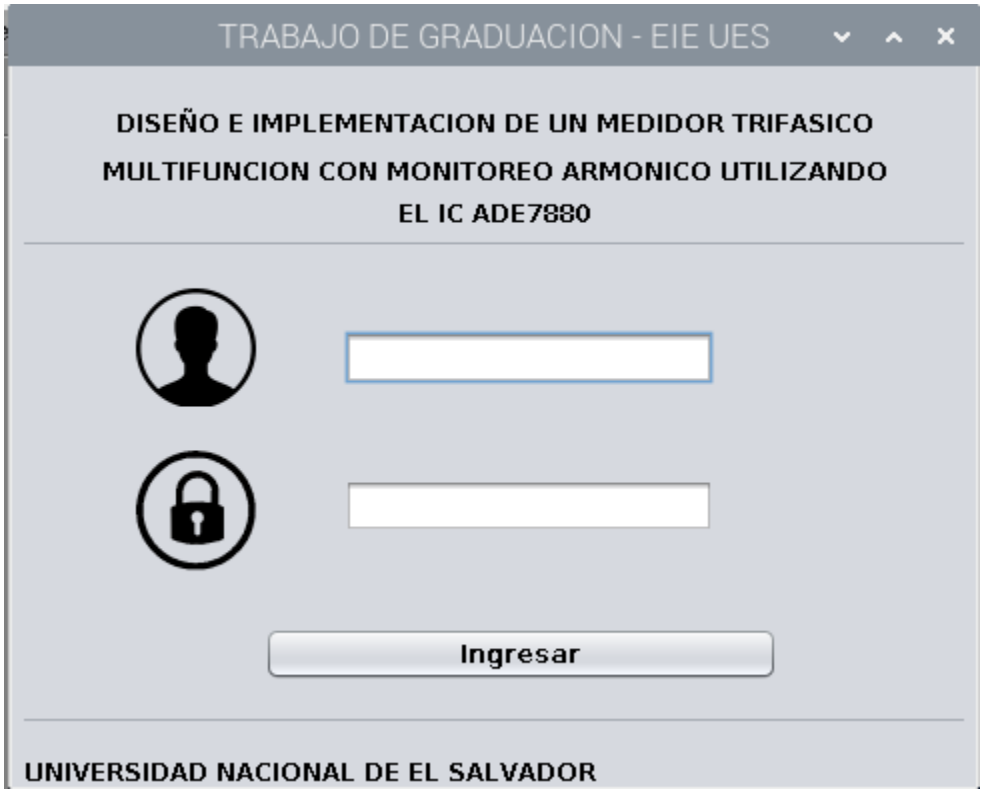


Figura 85: Interfaz gráfica de usuario para validación de sesiones.

3.1.4 Autenticación de sesiones

El código de autenticación se ejecuta luego de ingresar las credenciales y de que se presione el botón ingresar en la interfaz gráfica de usuario. La clase desarrollada se llama autenticación y en las figuras 86 y 87 se explican las partes más importantes de esta clase.

```
180         UsuariosJpaController Cusuario = new UsuariosJpaController();
181         List<Usuarios> usu;
182         usu = Cusuario.findUsuariosEntities();
183         String comparausu = jTextusuario.getText();
184         String comparacla = jPasswordusuario.getText();
```

Figura 86: Código de autenticación de sesiones variables y objetos utilizados.

En la línea 180 se crea un objeto del tipo usuariosjpacontroller y le se le da el nombre de Cusuario, este objeto se utiliza para la conexión con la unidad de persistencia y proporciona acceso a los métodos para crear, editar, eliminar y buscar. En la línea 181 y 182 se crea un objeto del tipo lista de la clase entidad usuarios. Una lista ordenada de los items que

conforman la tabla contenida en la base de datos. En las líneas 183 y 184 se declaran las variables para poder leer el jtexfield y el jpasswordfiel respectivamente. Aquí se capturan las entradas de texto.

```
186             int usua = 1, k = 0;
187             while( usua != 0 ){//comprueba que el usuario existe
188                 if( ( k + 1 )  == ( usu.size() + 1 ) ) {
189                   JOptionPane.showInternalMessageDialog(null,
190                           "usuario / contraseña incorrectos");
191                 }
192                 if (comparausu.equals(usu.get(k).getUsuario()) &&
193                     comparacla.equals(usu.get(k).getClave())){
194                     usua = 0;
195                 }
196                   k++;
197             }
198             System.out.println("\n\tOK USUARIO\t:" + usu.get(k - 1).getUsuario());
199             //posicionarme en el ususario y comprobar la bandera
200             if( usu.get(k - 1).getBandera() == 1 ){
201                 OpcionesAdmin pantalla2 = new OpcionesAdmin();
202                 pantalla2.setVisible(true);
203                 dispose();
204             }
205             if( usu.get(k - 1).getBandera() == 0 ) {
206                 MuestraDatos pantalla6 = new MuestraDatos();
207                 pantalla6.setVisible(true);
208                 dispose();
209             }
210
211     }
```

Figura 87: Código de autenticación de sesiones comprobación de usuarios.

En la figura 87 se muestra la parte del código que ejecuta la comprobación de los usuarios. En la línea 186 se declaran dos variables de tipo entero. En la línea 187 el lazo while y los dos if que contiene comparan los nombres de usuarios y las claves. En la línea 200 y 205 se comprueba la bandera asociada al usuario si es afirmativa el usuario tiene privilegios de administrador de lo contrario será un usuario de consulta. En las figuras 88 y 89 se muestran los dos tipos de acceso.

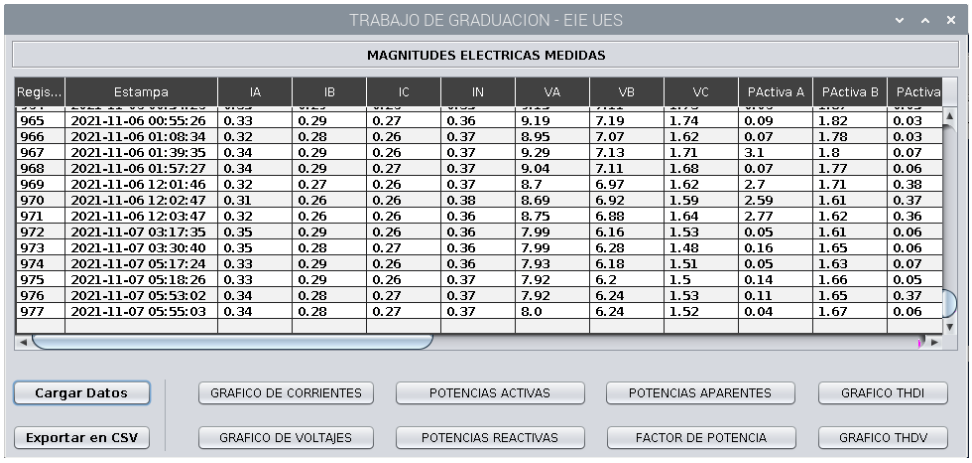


Figura 88: Acceso de autenticación como usuario para consulta.

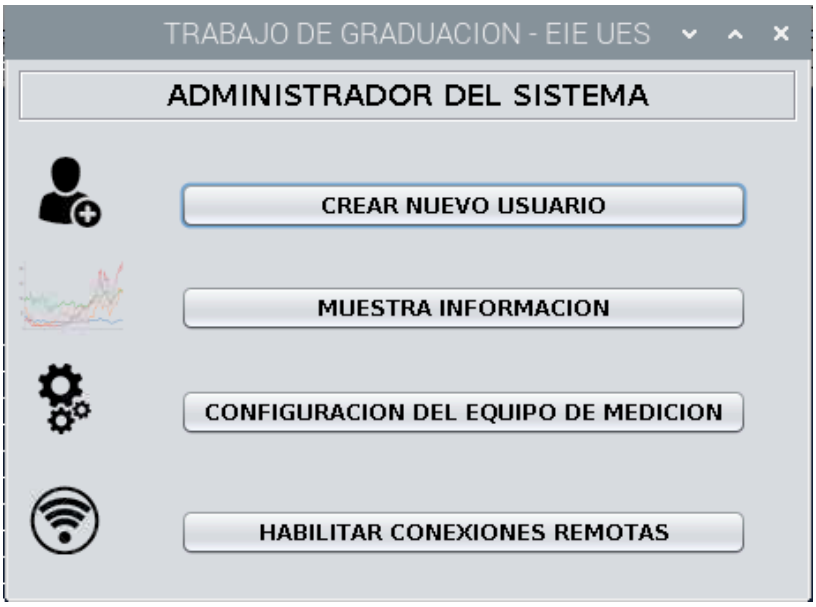


Figura 89: Acceso de autenticación como administrador.

3.2 Configuración y Obtención de las mediciones

El ADE7880 necesita ser configurado antes de quedar operativo en el equipo de medición. El proceso de configuración se ha desarrollado en métodos los cuales configuran los registros internos en memoria del ADE7880 para su óptima operación. La clase que ejecuta la configuración se llama mainade7880 y la base de datos donde están contenidas las tablas para el almacenamiento de las mediciones se llama MEDIDOR_TRIFASICO.BD y la tabla es MEDICONESCOMPLETAS. Para configurar y desarrollar el proceso de extracción de mediciones es necesario utilizar el puerto de comunicaciones I2C. Para establecer comunicación desde la clase es necesario utilizar algunos paquetes que nos permitan poder interactuar con los puertos de interés que para el presente trabajo son. El puerto de comunicaciones I2C y el puerto de propósito general GPIO soportado por la raspberry pi 4. En la figura 90 se muestran los paquetes necesarios para que la clase mainade7880 funcione correctamente.

```
6   package com.tesis2020.medidorade7880;
7
8
9
10  import java.io.IOException;
11  import io.dvlopt.linux.i2c.*;
12  import io.dvlopt.linux.gpio.*;
13  import static java.lang.Math.abs;
14  import static java.lang.Math.pow;
15  import static java.lang.Math.sqrt;
16  import java.sql.Timestamp;
17  import java.util.Date;
18  import java.text.SimpleDateFormat;
```

Figura 90: Paquetes de la clase Mainade7880.

En la línea 10 se tiene el paquete para controlar las excepciones al utilizar los puertos de comunicación. En la línea 11 se muestra el paquete para establecer comunicación por el puerto I2C. Cabe mencionar que existen un grupo de paquetes que son utilizados para controlar la comunicación vía I2C el que se ha seleccionado es io.dvlopt.linux.i2c. Porque nos proporciona las herramientas necesarias. En la línea 12 se hace referencia al paquete que da el soporte para acceder al puerto GPIO. Específicamente a los pines de E/S digital. En las líneas de la 13 a la 15 se muestra las funciones matemáticas necesarias. En las líneas de la 16 a la 18 son paquetes para el formato del timestamp, es decir el formato de fecha y hora que se utilizara para ser almacenado en la base de datos al extraer las mediciones. Para la configuración del ADE7880 se han desarrollado tres métodos en la clase mainade7880. Estos métodos cumplen la función de habilitar la escritura en la memoria RAM interna del DSP en el CI ADE7880, otro método que deshabilita la escritura en la memoria RAM interna del DSP en el CI ADE7880 y el un tercer método que configura los registros internos en la memoria RAM interna del DSP contenida en el ADE7880.

3.2.1 Configuración por software del medidor trifásico ADE7880

En la figura 91 se muestra el diagrama general del orden de ejecución de los métodos para generar la configuración inicial del CI ADE7880.

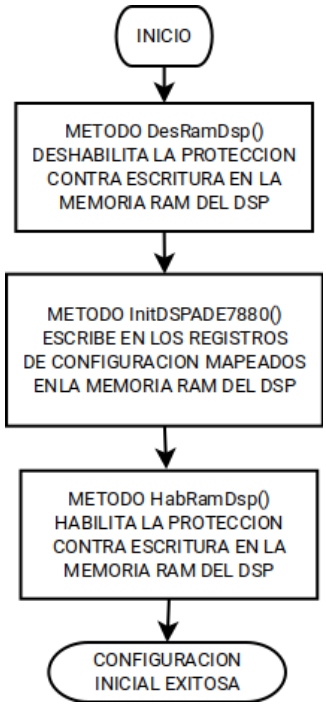


Figura 91: Métodos para configurar el CI ADE7880.

En las figuras 92, 94 y 95 se presenta el código correspondiente a cada método y se explican las partes más importantes.

```
236  private static void DesRamDsp() {
237
238          I2CBuffer Dir_Hab1 = new I2CBuffer( 3 ).set( 0, REG_DSP1H )
239                                                 .set( 1, REG_DSP1L )
240                                                 .set( 2, DATO_DSP1 );
241          I2CBuffer Dir_Hab2 = new I2CBuffer( 3 ).set( 0, HAB_RAM_DSPH )
242                                                 .set( 1, HAB_RAM_DSPL )
243                                                 .set( 2, DATO_DES_RAM_DSP );
244    try {
245          I2CBus  i2c = new I2CBus("/dev/i2c-1");
246
247          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
248          i2c.write( Dir_Hab1 );
249          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
250          i2c.write( Dir_Hab2 );
251          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
252          i2c.write( Dir_Hab2 );
253          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
254          i2c.write( Dir_Hab2 );
255          System.out.println( "\t\t" + "Echo DES PROT RAM DSP" + "\n");
256          i2c.close();
257    }
258    catch (IOException ex) {
259          Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
260      }
261  }
```

Figura 92: Deshabilitar la protección contra escritura memoria RAM del DSP.

En la línea 238 se declara un objeto del tipo I2CBuffer el cual proporciona la capacidad de enviar y recibir datos de 8bits. En la declaración del objeto tipo I2CBuffer se han definido 3 bytes lo que corresponde a 24bits. Los registros del ADE7880 están en el rango de los 8, 16, 24 y 32 bits por lo que son habituales las declaraciones de objetos de 1, 2, 3 y 4 bytes. La forma de funcionar de los objetos tipo I2CBuffer es similar a la de un arreglo de memoria. En la línea 244 está situada la sentencia try esta sentencia es necesaria debido a que se está trabajando con un puerto de comunicaciones y pueden surgir excepciones durante la ejecución del programa, por lo que con las sentencias try y catch se hace el manejo adecuado de las posibles excepciones de E/S en el puerto I2C. En la línea 245 se declara un objeto del tipo I2CBus el cual recibe como parámetro la dirección del fichero que controla el puerto I2C que para este caso en particular es en el tradicional directorio /dev/i2c-1. En las líneas 247 y 248 se ejecuta la operación de escritura en los registros de memoria del DSP. Se puede observar que en la línea 247 se envía la dirección del esclavo al bus I2C, luego en la línea 248 se escribe pero esta escritura tiene la siguiente lógica. El objeto I2CBuffer Dir_Hab1 se ha definido para controlar 3 bytes en los bytes más significativos se encuentra la dirección en hexadecimal del registro que se necesita escribir y en el byte menos significativo se encuentra el dato a escribir en el registro. Los registros han sido definidos como globales en la clase, como se muestra en la figura 94. En la línea 37 se define la variable de tipo byte ADE7880_ADDR_I2C que contiene la dirección del puerto I2C del CI ADE7880. Sin

embargo en las líneas 42 y 43 se muestran dos variables del tipo Byte la unión de estas dos variables representan la dirección de memoria del registro GAIN la cual es 0XE228. Es por este motivo que en los objetos tipo I2CBuffer siempre están en los registro 0 y 1 las variables tipo byte REG_NOMBRE_DEL_REGISTRO, por que contienen la dirección del registro de interés.

```
37    public static final byte ADE7880_ADDR_I2C = (byte)0x38;
38
39    public static final byte REG_LAST_OPH = (byte)0xEA;
40    public static final byte REG_LAST_OPL = (byte)0x01;
41
42    public static final byte REG_RUNH = (byte)0xE2;
43    public static final byte REG_RUNL = (byte)0x28;
```

Figura 93: Declaración de direcciones en memoria de los registros.

Siguiendo con el método de la figura 94. En la línea 256 se cierra el puerto I2C. En las líneas 258 y 259 esta implementado el catch para alguna eventual excepción de E/S y con esto se ha deshabilitado la protección contra escritura de la memoria RAM del DSP, por lo que ahora se puede escribir en los registro internos del DSP. Para ello se llama al método InitDSPADE7880 y su código se muestra en la figura 95. En las figuras 61 y 62 se encueran los diagramas de flujo para el proceso de lectura y escritura de registros del ADE7880 si se necesita tener mayor claridad del proceso.

```
262  private static void initDSPADE7880() {
263
264          I2CBuffer DirCOMPMODE_W = new I2CBuffer( 4 ).set( 0, REG_COMPMODEH ).set( 1, REG_COMPMODEL )
265                                                      .set( 2, DATO_COMPMODEH ).set( 3, DATO_COMPMODEL );
266          I2CBuffer DirCONFIG2_W = new I2CBuffer( 3 ).set( 0, REG_CONFIG2H ).set( 1, REG_CONFIG2L )
267                                                     .set( 2, DATO_CONFIG2 );
268          I2CBuffer DIrVLEVEL_W = new I2CBuffer( 6 ).set( 0, REG_VLEVELH).set( 1, REG_VLEVELL)
269                                                    .set( 2, DATO_VLEVELH1).set( 3, DATO_VLEVELH2)
270                                                    .set( 4, DATO_VLEVELL1).set( 5, DATO_VLEVELL2);
271          I2CBuffer DirCFMODE_W = new I2CBuffer( 4 ).set( 0, REG_CFMODEH ).set( 1, REG_CFMODEL )
272                                                    .set( 2, DATO_CFMODEH1 ).set( 3, DATO_CFMODEL1 );
273          I2CBuffer DirCF1DEN_W = new I2CBuffer( 4 ).set( 0, REG_CF1DENH ).set( 1, REG_CF1DENL )
274                                                    .set( 2, DATO_CF1DENH1 ).set( 3, DATO_CF1DENL1 );
275          I2CBuffer DirDSP_W = new I2CBuffer( 4 ).set( 0, REG_RUNH ).set( 1, REG_RUNL )
276                                                 .set( 2, DATO_RUNH ).set( 3, DATO_RUNL );
277     try {
278          I2CBus  i2c = new I2CBus("/dev/i2c-1");
279
280          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
281          i2c.write( DirCONFIG2_W );
282          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
283          i2c.write( DirCOMPMODE_W);
284          i2c.selectSlave( ADE7880_ADDR_I2C );
285          i2c.write( DIrVLEVEL_W );
286          i2c.selectSlave( ADE7880_ADDR_I2C );
287          i2c.write( DirCFMODE_W );
288          i2c.selectSlave( ADE7880_ADDR_I2C );
289          i2c.write( DirCF1DEN_W );
290          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
291          i2c.write( DirDSP_W );
292          i2c.selectSlave( ADE7880_ADDR_I2C ) ;
293          i2c.write( DirDSP_W );
294          i2c.selectSlave( ADE7880_ADDR_I2C );
295          i2c.write( DirDSP_W );
296          System.out.println( "\t\t" + "Iniciando la DSP del ADE7880" + "\n");
```

Figura 94: Escribir en los registros de configuración del ADE7880.

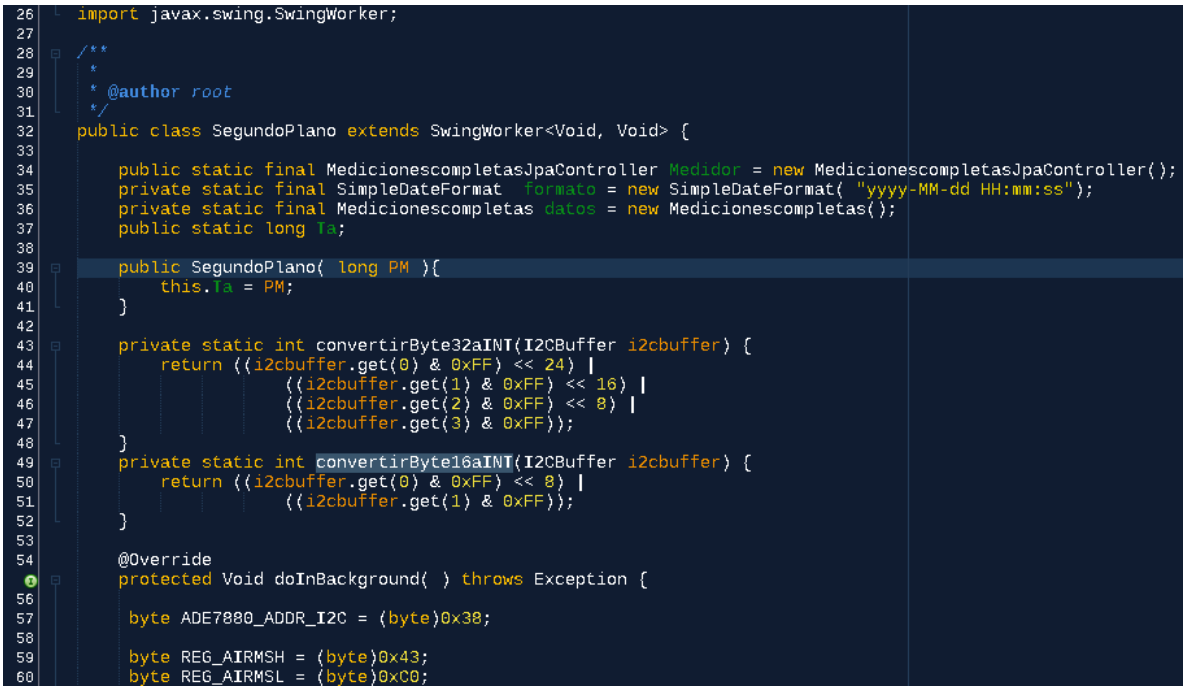
El método de la figura 95 sigue la misma lógica que el método de la figura 93. La parte más importante es la declaración de los objetos I2CBuffer. Que van de la línea 264 a la 275 se puede observar que la mayoría de registros de configuración son de 4bytes. En la figura 95 se presenta el código para habilitar la protección contra escritura de la memoria RAM del DSP. Al finalizar la ejecución de los tres métodos ya se tiene configurado correctamente el CI ADE7880 para entrar en operación permanente.

```
209     private static void HabRamDsp() {
210
211             I2CBuffer Dir_Hab1 = new I2CBuffer( 3 ).set( 0, REG_DSP1H )
212                                                    .set( 1, REG_DSP1L )
213                                                    .set( 2, DATO_DSP1 );
214             I2CBuffer Dir_Hab2 = new I2CBuffer( 3 ).set( 0, HAB_RAM_DSPH )
215                                                    .set( 1, HAB_RAM_DSPL )
216                                                    .set( 2, DATO_HAB_RAM_DSP );
217         try {
218             I2CBus  i2c = new I2CBus("/dev/i2c-1");
219
220             i2c.selectSlave( ADE7880_ADDR_I2C ) ;
221             i2c.write( Dir_Hab1 );
222             i2c.selectSlave( ADE7880_ADDR_I2C ) ;
223             i2c.write( Dir_Hab2 );
224             i2c.selectSlave( ADE7880_ADDR_I2C ) ;
225             i2c.write( Dir_Hab2 );
226             i2c.selectSlave( ADE7880_ADDR_I2C ) ;
227             i2c.write( Dir_Hab2 );
228             System.out.println( "\t\t" + "Echo HAB PROT RAM DSP" + "\n");
229             i2c.close();
230          }
231             catch (IOException ex) {
232             Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
233         }
234     }
```

Figura 95: Habilitar la protección contra escritura memoria RAM del DSP.

3.2.2 Obtención de las mediciones del ADE7880

El método desarrollado para la obtención de las mediciones en forma periódica se ejecuta en segundo plano y las mediciones se hacen cada 5 minutos. Por medio de este método se tiene acceso a las lecturas de los registros donde se almacenan las mediciones de corrientes, voltajes, potencias y distorsión armónica para corrientes y voltajes. El paquete java que proporciona herramientas para poder trabajar en segundo plano es swingworker con este paquete base se crea la clase segundoplano la cual contiene el método para la adquisición de mediciones, se realizan las conversiones y los cálculos con las mediciones obtenidas, se les da a las mediciones ya calculadas un formateo antes de ser almacenadas en la respectiva base de datos. La estructura básica de una clase que haga uso de la ejecución en segundo plano con el paquete swingworker es como se muestra en la figura 96.



```
26  import javax.swing.SwingWorker;
27
28  /**
29   *
30   * @author root
31   */
32  public class SegundoPlano extends SwingWorker<Void, Void> {
33
34      public static final MedicionescompletasJpaController Medidor = new MedicionescompletasJpaController();
35      private static final SimpleDateFormat  formato = new SimpleDateFormat( "yyyy-MM-dd HH:mm:ss");
36      private static final Medicionescompletas datos = new Medicionescompletas();
37      public static long Ta;
38
39      public SegundoPlano( long PM ){
40          this.Ta = PM;
41      }
42
43      private static int convertirByte32aINT(I2CBuffer i2cbuffer) {
44          return ((i2cbuffer.get(0) & 0xFF) << 24) |
45                      ((i2cbuffer.get(1) & 0xFF) << 16) |
46                      ((i2cbuffer.get(2) & 0xFF) << 8) |
47                      ((i2cbuffer.get(3) & 0xFF));
48      }
49      private static int convertirByte16aINT(I2CBuffer i2cbuffer) {
50          return ((i2cbuffer.get(0) & 0xFF) << 8) |
51                      ((i2cbuffer.get(1) & 0xFF));
52      }
53
54      @Override
        protected Void doInBackground( ) throws Exception {
56
57       byte ADE7880_ADDR_I2C = (byte)0x38;
58
59       byte REG_AIRMSH = (byte)0x43;
60       byte REG_AIRMSL = (byte)0xC0;
```

Figura 96: Clase SegundoPlano.

En la línea 26 de la figura 96 se muestra la importación del paquete swingworker. En la línea 32 se declara la clase con la palabra reservada extends esta palabra reservada tiene la función de hacer una espacie de herencia de los métodos y atributos de la clase swingworker a la clase sengundoplano. En la línea 39 se declara el constructor de la clase el cual da la funcionalidad de recibir un parámetro o atributo de tipo long por medio del cual se pasa el tiempo de adquisición de mediciones. En la línea 55 se declara el método doInbackground y es en donde se ejecuta nuestro código para la obtención de las mediciones programando un periodo de tiempo que se repite mientras se esté haciendo uso del sistema de medición. Estas partes engloban la estructura básica de la clase segundoplano. En las líneas 43 hasta la 48 está el método converirbyte32aint el cual convierte las lectura que contiene el objeto i2cbuffer de cuatro bytes a un número entero. De las líneas 49 a la 52 se muestra el método converirbyte16aint el cual convierte las lecturas de un objeto i2cbuffer de dos bytes a un número entero. En la línea 34 se declara un objeto de la clase controladora medicionescompletasjpacontroller. En la línea 35 se declara un objeto de tipo simpledateformat el cual será usado para obtener la fecha y hora en el formato requerido. En la línea 36 se crea un objeto de la clase entidad medicionescompletas y en la línea 37 se declare el atributo Ta de tipo long que se usa para poder dar el tiempo de adquisición de mediciones. La lógica de funcionamiento del método doInbackground se muestra en la figura 97.

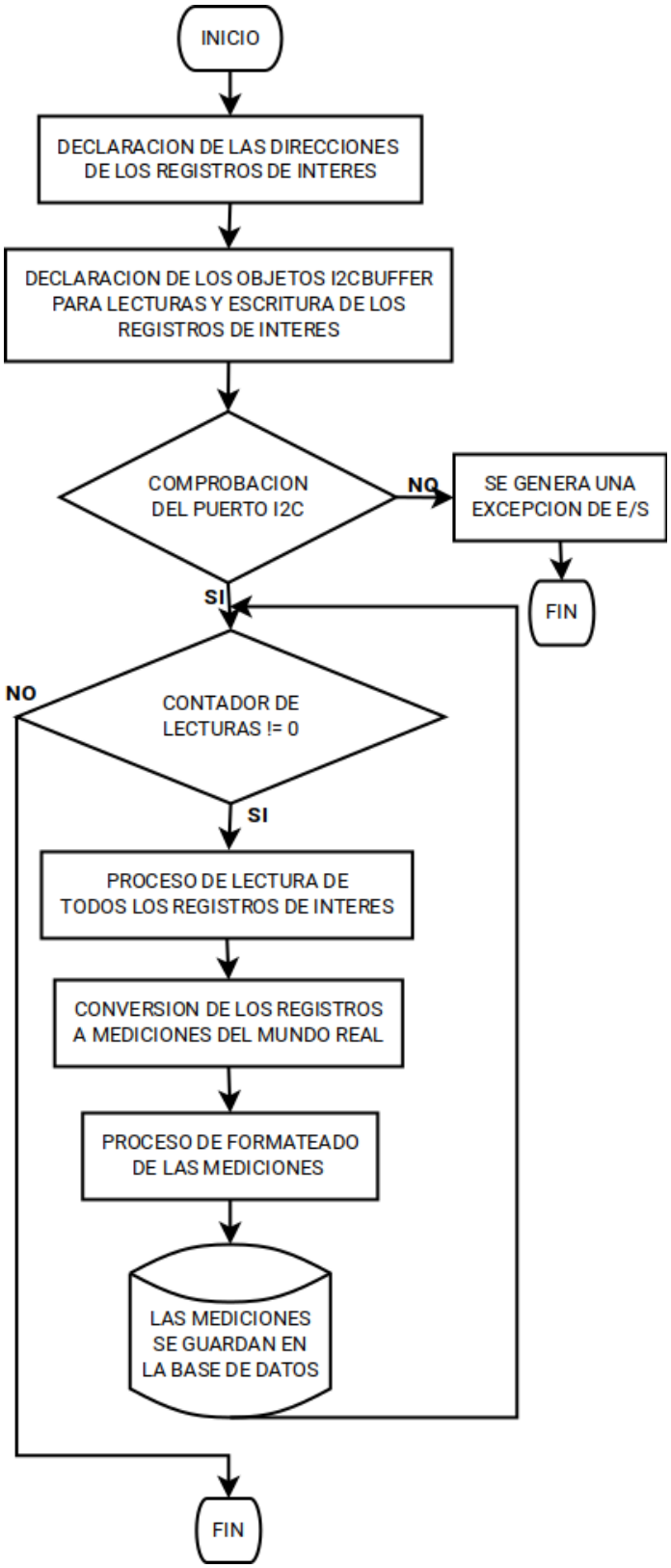


Figura 97: Lógica de ejecución del método doInbackground.

3.3 Indicadores por el puerto GPIO

Los métodos por medio de los cuales se puede hacer uso de los pines de entra y salida del puerto GPIO están implementados en la clase configeqmed así como la declaración de los atributos necesarios para su implementación. Se hace uso de cuatro métodos para el control de los indicadores del equipo de medición, dos de ellos se muestran en la figura 98.

```
185     private static void indicadores( GpioHandle handle,
186                                      GpioBuffer buffergpio,
187                                      GpioLine controlpin,
188                                      boolean salidaPin ) {
189     try {
190          buffergpio.set(controlpin, salidaPin);
191          handle.write(buffergpio);
192     }
193     catch (IOException ex) {
194             Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
195         }
196      }
197
198     private static void encendido( boolean TF ){
199         try {
200             GpioHandleRequest reques = new GpioHandleRequest().setConsumer("gpioade7880")
201                                              .setFlags(new GpioFlags().setOutput());
202
203             GpioLine encendido = reques.addLine( PIN_ENCENDIDO, true );
204             GpioBuffer buffergpio = new GpioBuffer();
205             GpioDevice device_gpio = new GpioDevice( GPIO_DEVICE );
206             GpioHandle handle = device_gpio.requestHandle(reques);
207             indicadores( handle, buffergpio, encendido, TF );
208             device_gpio.close();
209             handle.close();
210
211             System.out.println("\n\t" + "indicaror encendido:" + encendido + "\n");
212     }
213     catch( IOException ex){
214         Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
215         }
216
217     }
```

Figura 98: Métodos indicadores y encendido por medio del puerto gpio.

El paquete necesario para el control del puerto GPIO es gpio. En la línea 185 se declara el método indicadores que es de uso común para los otros tres métodos. Acá se declaran los parámetros que recibe el método. Los cuales son tres objetos y una variable de tipo booleana. En la línea 190 al objeto tipo gpiobuffer se le asignan dos valores el primero es el número del pin de E/S y el segundo es el estado lógico uno o cero. En la línea 191 el objeto de tipo gpiohandle escribe en el pin especificado un valor lógico que puede ser un uno lógico o un cero lógico. El método encendido en la línea 198 recibe un parámetro el cual contiene el estado lógico del pin seleccionado. En la línea 200 se declara un objeto del tipo gpiohandlerequest el cual nos permite seleccionar el pin que se desea controlar. En la línea 203 se crea un objeto del tipo gpioline el cual recibe la asignación del pin de interés por medio del objeto reques. La línea 204 crea un objeto del tipo gpiobuffer. En la línea 205 se crea un objeto del tipo gpiodevice al cual se le asigna la dirección del fichero que controla el

puerto GPIO. En un sistema GNU/LINUX la dirección es /dev/gpiochip0. En la línea 206 se crea un objeto del tipo gpiohandle el cual adquiere el control del puerto por medio de los objetos device_gpio y request. En la línea 207 se llama al método indicadores para que este escriba en el puerto GPIO. En las líneas 208 y 209 se cierra el puerto GPIO. Los otros dos métodos se ejecutan de la misma forma descrita acá. Con la salvedad de que en estos otros métodos lo único que se cambia es el pin de interés. En la figura 99 se muestra la declaración de las variables que representan a los pines del puerto GPIO que se usan en los métodos.

```
22  public class ConfigEqmed extends javax.swing.JFrame {
23
24      public static long PME;
25
26      public static final int PIN_ENCENDIDO     = 17 ;
27      public static final int PIN_CONEXION_E    = 27 ;
28      public static final int PIN_CONEXION_D    = 22 ;
29      public static final String GPIO_DEVICE    = "/dev/gpiochip0";
```

Figura 99: variables necesarios para los métodos de control GPIO.

3.4 Creación de las bases de datos

Las bases de datos utilizadas por el equipo de medición son dos una que contiene una tabla con los usuarios registrados en el sistema y otra base de datos que contiene una tabla con los registros de las mediciones de los parámetros eléctricos. El gestor de base de datos relacional que se ha seleccionado es SQLITE. El proceso para crear las bases de datos es el siguiente: se abre un terminal y se digita el siguiente comando sqlite3 NOMBRE_BASE_DE_DATOS_NUEVA.DB ya con esto sea creado la base de datos y se entra en el CLI de sqlite ahora se procede a la creación de las tablas, esto se consigue con el siguiente comando CREATE TABLE NOMRE_DE_LA_TABLA (DECLARACION DE LOS TIPOS DE REGISTROS); las bases de datos y las tablas de uso en el equipo de medición son:

1. SESIONES.DB la cual contiene una tabla llamada USUARIOS.

2. MEDIDOR_TRIFASICO.DB la cual contiene una tabla llamada MEDICIONESCOMPLETAS.

Los tipos de datos de la tabla USUARIOS y la tabla MEDICIONESCOMPLETAS se muestran en las tablas 8 y 9 respectivamente.

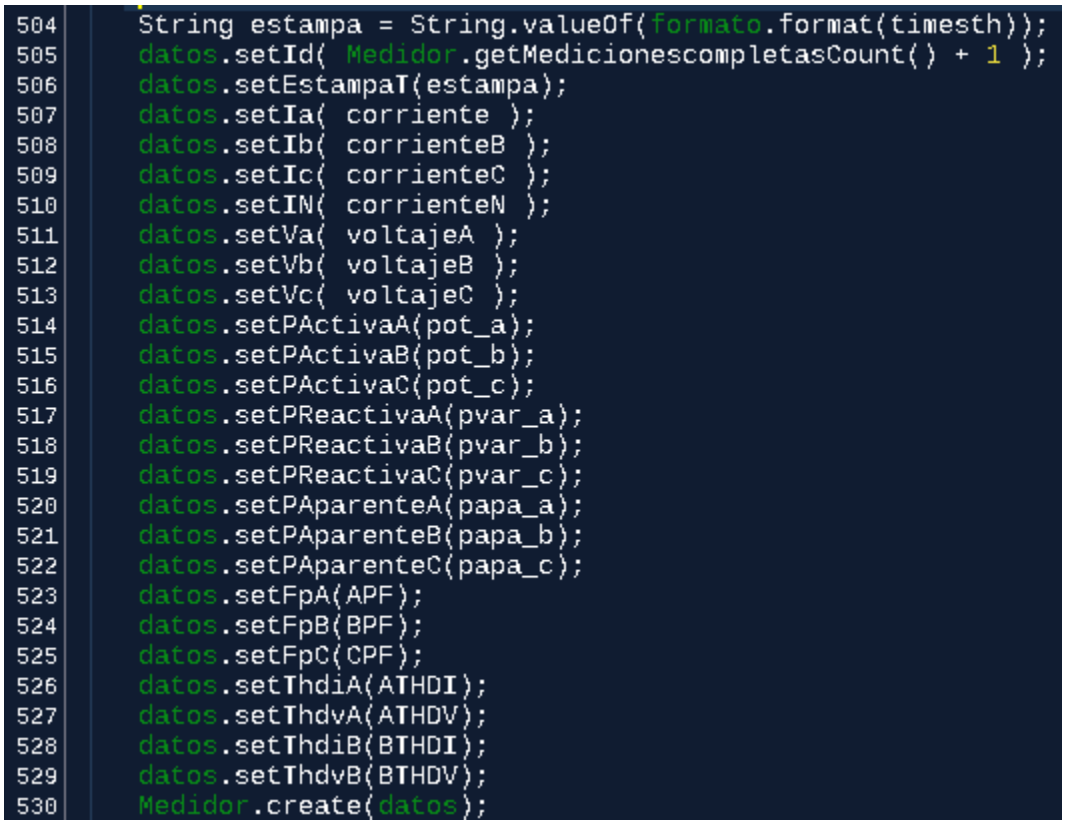
ENCABEZADOS	TIPOS DE REGISTROS
ID	INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
ESTAMPA	TEXT DEFAULT CURRENT_TIMESTAMP
USUARIO	TEXT DEFAULT 0
CLAVE	TEXT
BANDERA	BOOLEAN DEFAULT 0
PRIMARY KEY (ID));	

Tabla 8: Estructura de la tabla usuarios.

ENCABEZADOS	TIPO DE REGISTROS
ID	INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
ESTAMPA_T	TEXT DEFAULT CURRENT_TIMESTAMP
IA, IB e IC	DOUBLE DEFAULT 0
VA, VB y VC	DOUBLE DEFAULT 0
P_ACTIVA_A, B y C	DOUBLE DEFAULT 0
P_REACTIVA_A, B y C	DOUBLE DEFAULT 0
P_APARENTE_A, B y C	DOUBLE DEFAULT 0
FP_A, FP_B y FP_C	DOUBLE DEFAULT 0
THDI_A, THDI_B y THDI_C	DOUBLE DEFAULT 0
THDV_A, THDV_B y THDV_C	DOUBLE DEFAULT 0
PRIMARY KEY (ID));	

Tabla 9: Estructura de la tabla medicionescompletas.

El código java utilizado para almacenar las mediciones en la base de datos es el que se muestra en la figura 100 y se explican las partes más importantes de este.



```
504     String estampa = String.valueOf(formato.format(timesth));
505     datos.setId( Medidor.getMedicionescompletasCount() + 1 );
506     datos.setEstampaT(estampa);
507     datos.setIa( corriente );
508     datos.setIb( corrienteB );
509     datos.setIc( corrienteC );
510     datos.setIN( corrienteN );
511     datos.setVa( voltajeA );
512     datos.setVb( voltajeB );
513     datos.setVc( voltajeC );
514     datos.setPActivaA(pot_a);
515     datos.setPActivaB(pot_b);
516     datos.setPActivaC(pot_c);
517     datos.setPReactivaA(pvar_a);
518     datos.setPReactivaB(pvar_b);
519     datos.setPReactivaC(pvar_c);
520     datos.setPAparenteA(papa_a);
521     datos.setPAparenteB(papa_b);
522     datos.setPAparenteC(papa_c);
523     datos.setFpA(APF);
524     datos.setFpB(BPF);
525     datos.setFpC(CPF);
526     datos.setThdiA(ATHDI);
527     datos.setThdvA(ATHDV);
528     datos.setThdiB(BTHDI);
529     datos.setThdvB(BTHDV);
530     Medidor.create(datos);
```

Figura 100: Código java utilizado para almacenar las mediciones en la base de datos.

En la línea 504 la función String.valueof convierte el formato de hora y fecha a una cadena de caracteres y esta es asignada a la variable de tipo String estampa. En la línea 505 se usan dos objetos el primero es datos el cual es un objeto del tipo clase entidad medicionescommpletas y el segundo Medidor el cual es del tipo clase controladora medicionescomplestasjpacontroller, esta clase hace uso de la unidad de persistencia. Acá se obtiene el ultimo valor del registro ID de la base de datos y se le suma uno el resultado de esta operación es seteado en el registro ID. En la línea 506 el objeto datos permite tener acceso a todos los registros de la tabla y de esta forma se cambia el registro que almacena la fecha y hora. De la misma forma todos los demás registros de la tabla. En la línea 530 el objeto Medidor usa el método créate definido en la clase medicionescompletasjpacontroller para guardar en la base de datos las mediciones.

3.5 Interfaces graficas de usuario

Luego de haberse autenticado en una sesión de usuario se muestran una serie de ventanas o interfaces graficas desde las cuales se puede interactuar con los registros de las mediciones almacenados en la bases de datos. La interfaz creada en la clase muestradatos es la que permite llevar a cabo estas operaciones y pone a disposición del usuario una serie de botones

para generar gráficos de los registros de corrientes, voltajes, potencias, factor de potencia y de distorsión armónica. En la figura 101 se muestra la interface.

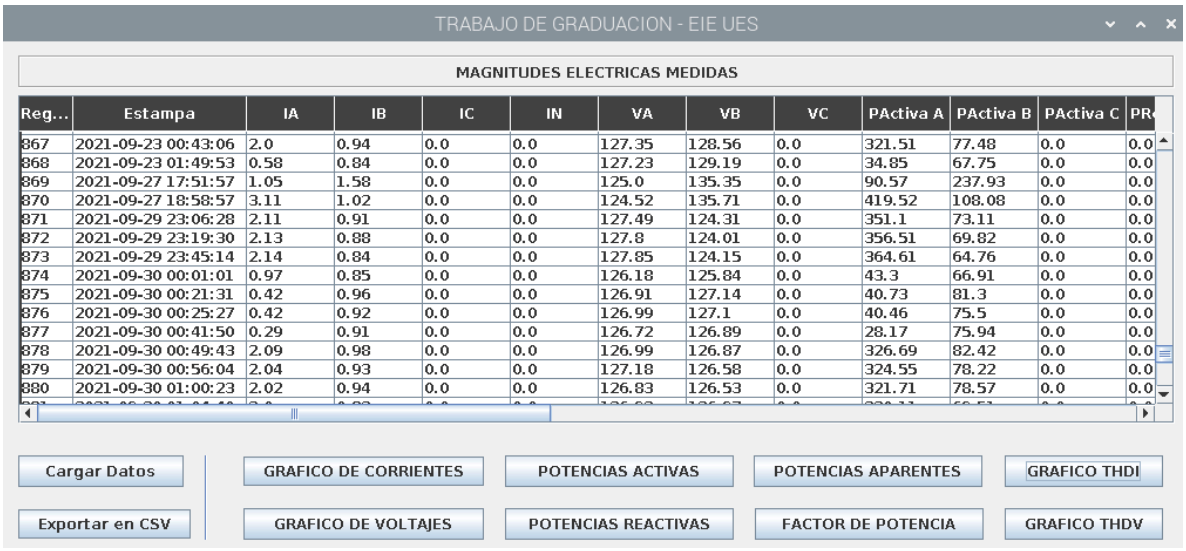


Figura 101: Interfaz gráfica de la clase MuestraDatos.

El botón cargar datos de la figura 101 ejecuta el método que lee todos los registros de la base de datos y los muestra en el modelo de tabla creado, las barras de desplazamiento horizontal y vertical dan la posibilidad de ubicar cualquier registro de medición que sea de interés. El botón exportar archivo en csv ejecuta el método para leer toda la base de datos y la guarda en un archivo con extensión csv para lo cual se hace uso del paquete opencsv. Los botones generan los gráficos a partir de los registros almacenados en la base datos y ejecutan los métodos para leer los registros de interés. Haciendo uso del paquete jfreechart se crean los gráficos correspondientes. En la figura 102 se presenta el diagrama de flujo para la creación de gráficas.

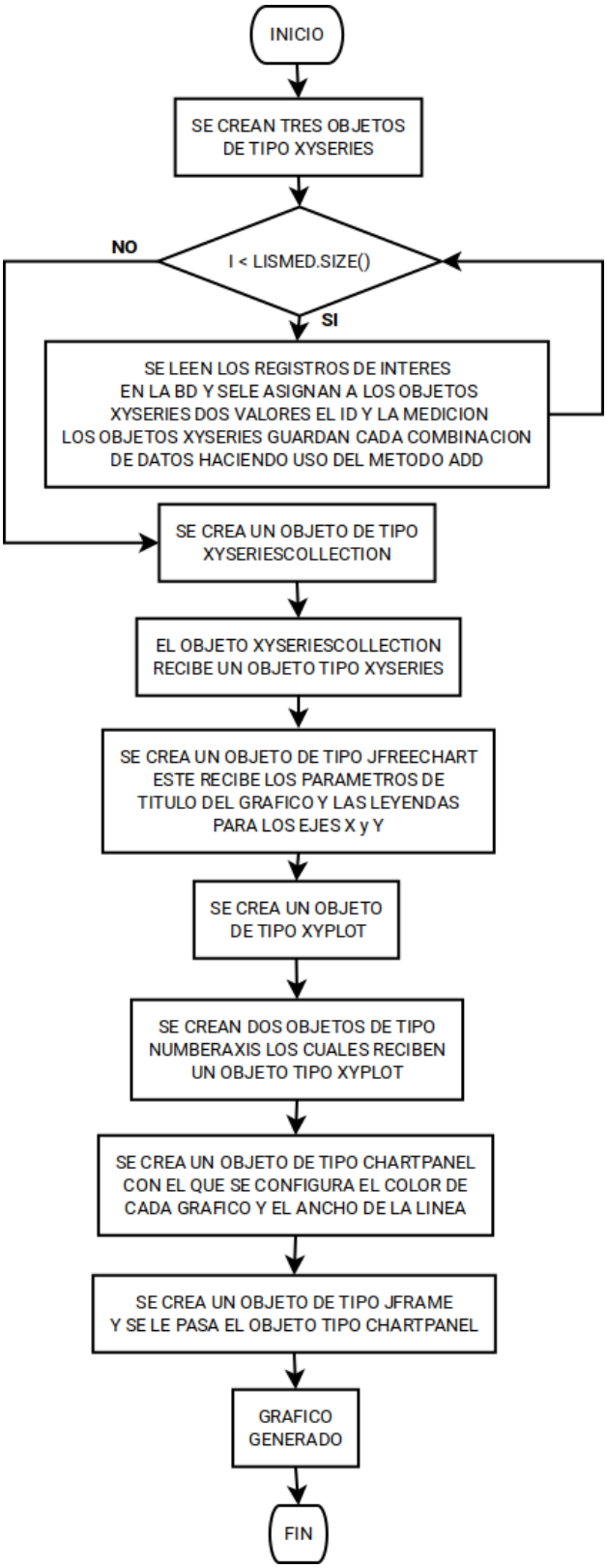


Figura 102: Diagrama de flujo para la creación de gráficos.

Otra interfaz gráfica de importancia es la del administrador del sistema. Esta se ha desarrollado en la clase opcionesadmin. En dicha clase se encuentran implementados los métodos necesarios para la configuración del equipo de medición así como la creación de nuevos usuarios o la eliminación de estos. Además en esta interfaz se encuentra la opción para habilitar o deshabilitar las conexiones vía remotas manipulando los comandos start y stop del servidor tomcat. Otra opción de configuración con la que se cuenta es con la posibilidad de habilitar o deshabilitar los indicadores del equipo de medición y finalmente se tiene una opción que cargar la interfaz gráfica contenida en la clase muestradatos. La interfaz gráfica del administrador se muestra en la figura 89. Las interfaces para cada opción se muestran en las figuras 103, 104 y 105.

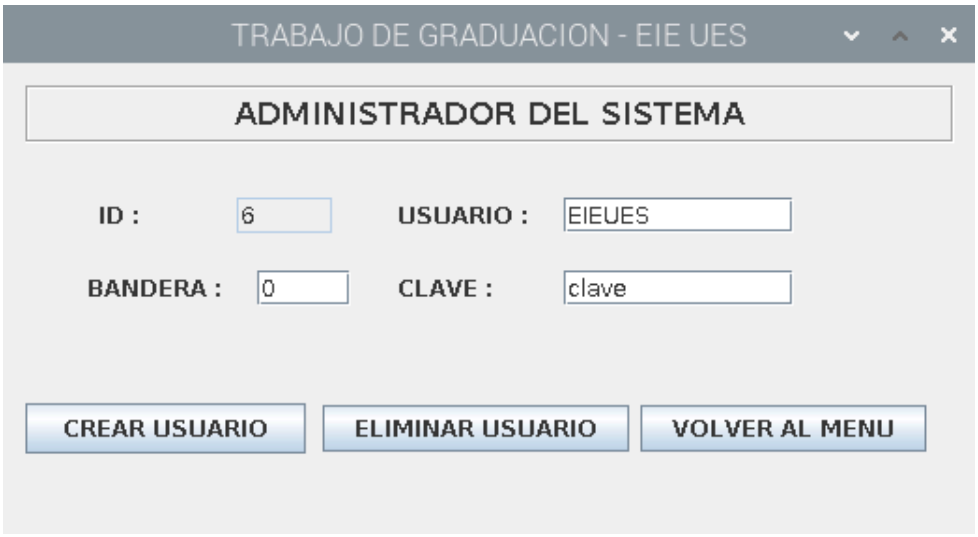


Figura 103: Interfaz gráfica para la opción crear nuevo usuario.

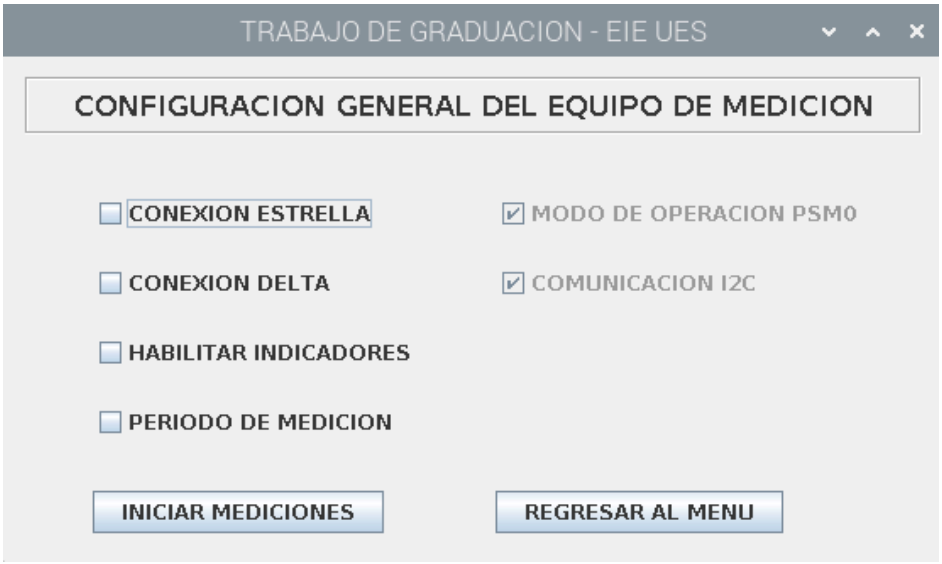


Figura 104: Interfaz gráfica para la opción configuración del equipo de medición.

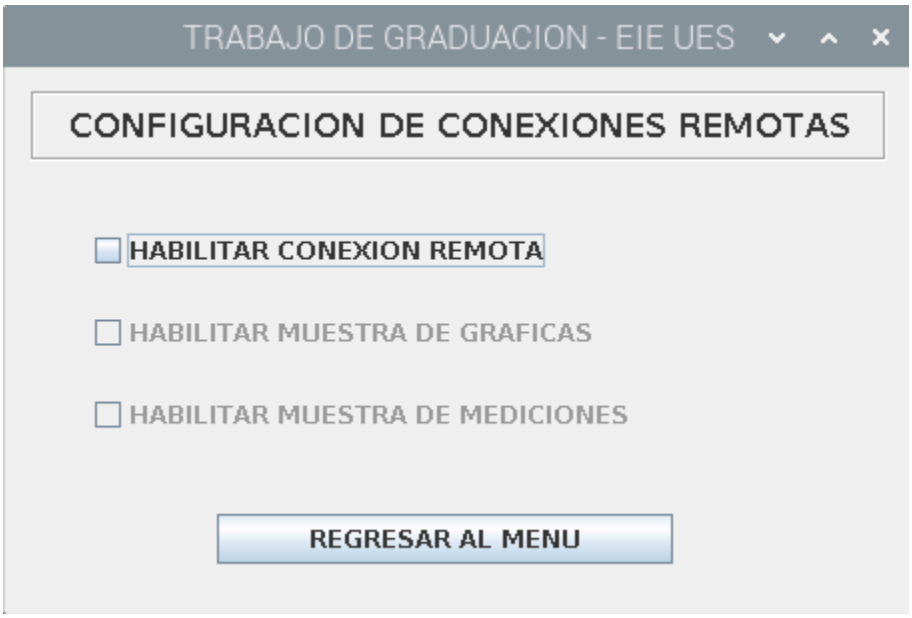


Figura 105: interfaz gráfica para la opción conexiones remotas.

3.6 Desarrollo del sitio web local

Para el desarrollo de esta parte se hará uso del servidor apache tomcat versión 9.0.46 y de la tecnología JSP para la creación de páginas web. La configuración del entorno de persistencia se hace según lo expuesto en el apartado 3.1.1. El servidor tomcat se encarga de servir las paginas jsp desarrolladas en una red de tipo LAN y el cliente es una aplicación android que corre sobre un teléfono inteligente. La forma en que se generan los gráficos es la misma expuesta en el flujo grama de la figura 102. La autenticación está habilitada y esta permite cargar los registros de las mediciones guardadas en la base de datos. Todo el desarrollo de software se hace en el IDE netbeans versión 12.3 el proyecto se llama Servidorade7880 el cual contiene todas las paginas jsp para la autenticación, carga de registros de mediciones en tabla y para la generación de gráficas. Esta es la parte que contiene el lado del servidor. La parte del cliente se desarrolló en el IDE Android-Studio versión 4.1.1 y consiste en la aplicación móvil.

3.6.1 Lado del servidor

El lado del servidor está formado por el software desarrollado y el sevidor apache tomcat. El software desarrollado consiste de todas las paginas jsp que maneja tomcat y las cuales ejecutan una tarea. La lógica de la autenticación es similar a la explicada en la sección 3.1 ver para más detalles aunque son entornos diferentes el lenguaje de programación es el

mismo por lo que con mínimas diferencias trabajan de forma similar. En el IDE netbeans se crea el archivo .WAR el cual contiene todo el proyecto listo para su ejecución vía web una vez que se ha creado este archivo se copia a la carpeta webapps en el sistema de ficheros de tomcat. Quedando listo para ser iniciado el servidor apache tomcat desde la interfaz gráfica del administrador, el método que inicia o detiene el servidor tomcat hace uso de los siguientes comandos systemctl start tomcat con este comando el servidor tomcat entra en operación. Para verificar el estado del servidor se utiliza el siguiente comando systemctl status tomcat y para detenerlo se usa el comando systemctl stop tomcat. Estos tres comandos son básicos para el control del servidor apache tomcat. Luego de haber iniciado el servidor tomcat se puede tener acceso al servicio web desarrollado. La dirección que se usa es de la forma genérica NUMERO_IP:NUMERO_PUERTO que se traduce en lo siguiente 1.1.0.4:8080 que significa la dirección IP donde corre el servidor y el puerto por donde escucha peticiones. El número de puerto por defecto para el servidor apache tomcat es el 8080. Todos los códigos de las clases desarrolladas se muestran en el anexo C. Las aplicaciones que contiene el archivo .WAR y que ejecuta el servidor tomcat son:

- Autenticación de usuario: permite tener acceso a la operación de cargar los registros de mediciones contenidos en la base de datos y listarlos en una tabla.
- Generación de gráficas para los distintos parámetros eléctricos medidos.

3.6.2 Lado del cliente

El cliente es la aplicación móvil. Opera de la siguiente forma cuando ya se tiene operando el lado del servidor se ejecuta la aplicación en un teléfono inteligente o tableta. En la figura 106 se muestra la aplicación funcionando. Los códigos desarrollados se muestran en el anexo D.

Figura 106: Aplicación móvil android.

Luego de autenticarse se puesden cargar los registros en la tabla de mediciones.

Registros	Estampa	IA	IB	IC	IN	VA	VB	VC	PActivaA	PActivaB	PActivaC	PReactivaA	PRe
1	2021-06-05 00:12:09	30.1	30.2	30.3	10.93	120.51	122.3	123.93	3445.98	3508.79	3567.33	1132.64	11
2	2021-06-12 18:47:14	2.35	0.0	0.0	0.0	123.03	0.0	0.0	244.16	0.0	0.0	0.0	

Figura 107: Cagar los registros de la base de datos en una tabla.

Una de las opciones para graficar las mediciones de los parámetros eléctricos se muestra en la figura 108.

Figura 108: Opción para generar grafico de corrientes.

Grafico generado de las corrientes de fase.

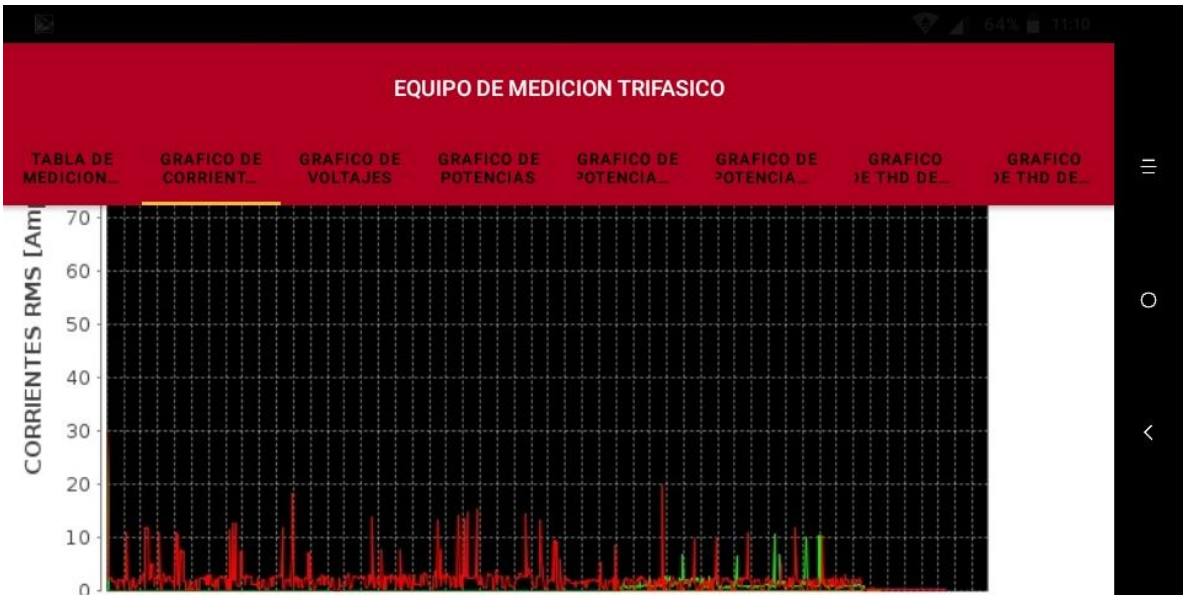


Figura 109: Grafico de corrientes de fases generado.

CONCLUSIONES

- El protocolo de comunicaciones I2C es una excelente opción para el transporte de información entre chips. Para sistemas embebidos. Ya que su uso práctico y simplicidad en su conexión lo vuelve una excelente solución de comunicación.

- El medidor ADE7880 es ideal para implementar medidores de energía eléctrica del tipo analizadores con la finalidad de obtener el máximo provecho del motor de cálculos de armónicos con el que cuenta, y las ventajas de calibración que ofrece con los registros específicos para compensar o amplificar las señales de entrada resultan practicas e importantes.

- La computadora de placa única raspberry pi es una alternativa de soporte con un costo/beneficio excelente. Todas las capacidades de procesamiento y de comunicación que proporciona, además de la potencia en la capacidad de desarrollo de software que presta la hacen una opción ideal en el desarrollo de sistemas embebidos.

- El uso de nuevas tecnologías como la impresión 3D, se hacen cada vez más imprescindibles al momento de desarrollar proyectos o sistemas que requieren de una carcasa para el montaje de estos. Por tanto el dominio del software de diseño así como del Hardware (impresora 3D) se vuelven importantes para la formación profesional de los futuros ingenieros.

- Las aplicaciones móviles son una herramienta muy importante ya que pueden proporcionan información relevante sin la necesidad de equipos costosos o de difícil movilidad.

RECOMENDACIONES

- Si se emplea el circuito integrado ADE7880 con encapsulado QFN en posteriores trabajos o desarrollos se recomienda comprarlo ya soldado en una PCB debido a que por la tecnología de encapsulado esta puede llegar hacer una ardua tarea.

- Los transformadores de corriente se deben elegir con las mejores características debido a que de ellos depende en gran medida la confiabilidad de las mediciones realizadas.

- En un diseño en el que se imprima en una impresora 3D, se debe tener en cuenta que estas siempre tienen un margen de error y se debe considerar en el modelo por software.

- El filtro paso alto digital en la trayectoria de la fase C del voltaje permanece habilitado sin importar la configuración en el registro CONFIG3.

REFERENCIAS

[1] Hoja de datos del circuito integrado ADE7880. [En línea]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ADE7880.pdf.[Último acceso: 20 Septiembre 2021]

[2] Notas de aplicación del circuito integrado ADE7880. [En línea]. Available: https://www.analog.com/media/en/technical-documentation/application-notes/.[Último acceso: 04 Octubre 2021]

[3] Zona de ingeniería, comunidad de soporte. [En línea]. Available: https://ez.analog.com/search?q=ADE7880.[Último acceso: 12 Septiembre 2021]

[4] Hoja de datos de los transformadores de corriente. [En línea]. Available: https://en.yhdc.com/product/SCT013-401.html.[Último acceso: 08 Junio 2021]

[5] Documentación de la placa raspberry pi. [En línea]. Available: https://www.raspberrypi.com/documentation/.[Último acceso: 14 Mayo 2021]

[6] Serie schaum circuitos eléctricos Joseph A. Edminister.2ED.

[7] José Dariel Arcila ARMÓNICOS EN SISTEMAS ELÉCTRICOS.pdf.

INGENIERÍA ESPECIALIZADA S.A.

[8] A.Tejada, A.Llamas EFECTOS DE LAS ARMÓNICAS EN LOS SISTEMAS ELÉCTRICOS.pdf

Departamento de Ingeniería Eléctrica del ITESM Campus Monterrey

[9] J. O. Murcia Flores, “Equipo concentrador de vatímetros para uso en el sector doméstico e industrial”, Universidad de El Salvador, San Salvador, El Salvador, 2014.

ANEXOS

Anexo A. Instalación del S.O. raspbian en una memoria microSD

- Descargar la imagen del sitio oficial raspbian https://downloads.raspberrypi.org/raspios_full_armhf/images/raspios_full_armhf-2021-11-08/2021-10-30-raspios-bullseye-armhf-full.zip
- Luego conectar a la computadora la memoria SD en la que se instalara el S.O, abrir el programa imagewriter y seleccionar la imagen descargada luego seleccionar la memoria SD y presionar el botón write. Como se muestra en la figura 110.

Figura 110: Aplicación para instalar raspbian.

- Cuando se haya terminado de instalar el S.O. en la memoria micro SD cerrar el programa y desmontar la memoria micro SD, posteriormente insertarla en la ranura respectiva de la tarjeta raspberry pi 4.
- Conectar la fuente de poder a la raspberry pi 4 y encender.

Anexo B. Hoja de datos del 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 $\overline{RST}$ pin is monitored as a pushbutton input for generating a µP reset.

Benefits and Features

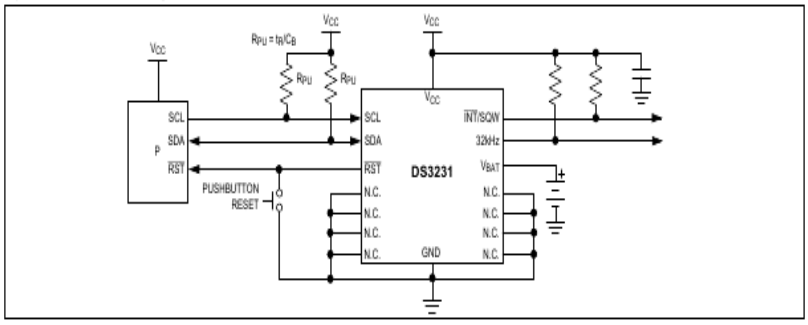
- Highly Accurate RTC Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
  - Accuracy ±2ppm from 0°C to +40°C
  - Accuracy ±3.5ppm from -40°C to +85°C
  - Digital Temp Sensor Output: ±3°C Accuracy
  - Register for Aging Trim
  - $\overline{RST}$ 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.

Typical Operating Circuit



Underwriters Laboratories is a registered certification mark of Underwriters Laboratories Inc.

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground....-0.3V to +6.0V
Junction-to-Ambient Thermal Resistance (θ_{JA}) (Note 1)73°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) (Note 1)....23°C/W
Operating Temperature Range
DS3231S .....0°C to +70°C
DS3231SN..... -40°C to +85°C
Junction Temperature.....+125°C
Storage Temperature Range..... -40°C to +85°C
Lead Temperature (soldering, 10s).....+260°C
Soldering Temperature (reflow, 2 times max) .....+260°C
(see the *Handling, PCB Layout, and Assembly* section)

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°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}	I2C bus inactive, 32kHz output on, SQW output off (Note 5)	V_{CC} = 3.63V			110	µA
			V_{CC} = 5.5V			170	
Temperature Conversion Current	$I_{CCSCONV}$	I2C 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, $\overline{INT}$/SQW, SDA	V_{OL}	I_{OL} = 3mA				0.4	V
Logic 0 Output, $\overline{RST}$	V_{OL}	I_{OL} = 1mA				0.4	V
Output Leakage Current 32kHz, $\overline{INT}$/SQW, SDA	I_{LO}	Output high impedance		-1	0	+1	µA
Input Leakage SCL	I_{LI}			-1		+1	µA
$\overline{RST}$ Pin I/O Leakage	I_{OL}	$\overline{RST}$ 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	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	ppm
			0°C to +40°C			±2	
			>40°C to +85°C			±3.5	
Frequency Stability vs. Voltage	Δf/V				1		ppm/V
Trim Register Frequency Sensitivity per LSB	Δf/LSB	Specified at:	-40°C		0.7		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		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}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = 400kHz (Note 5)	V_{BAT} = 3.63V			70	µA
			V_{BAT} = 5.5V			150	
Timekeeping Battery Current	I_{BATT}	$\overline{EOSC}$ = 0, BBSQW = 0, EN32kHz = 1, SCL = SDA = 0V or SCL = SDA = V_{BAT} (Note 5)	V_{BAT} = 3.63V		0.84	3.0	µA
			V_{BAT} = 5.5V		1.0	3.5	
Temperature Conversion Current	I_{BATTC}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = SDA = 0V or SCL = SDA = V_{BAT}	V_{BAT} = 3.63V			575	µA
			V_{BAT} = 5.5V			650	
Data-Retention Current	I_{BATTDR}	$\overline{EOSC}$ = 1, SCL = SDA = 0V, +25°C				100	nA

AC Electrical Characteristics

(V_{CC} = $V_{CC(MIN)}$ to $V_{CC(MAX)}$ or V_{BAT} = $V_{BAT(MIN)}$ to $V_{BAT(MAX)}$, V_{BAT} > V_{CC}, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}	Fast mode	100		400	kHz
		Standard mode	0		100	
Bus Free Time Between STOP and START Conditions	t_{BUF}	Fast mode	1.3			µs
		Standard mode	4.7			
Hold Time (Repeated) START Condition (Note 7)	$t_{HD:STA}$	Fast mode	0.6			µs
		Standard mode	4.0			
Low Period of SCL Clock	t_{LOW}	Fast mode	1.3			µs
		Standard mode	4.7			
High Period of SCL Clock	t_{HIGH}	Fast mode	0.6			µs
		Standard mode	4.0			
Data Hold Time (Notes 8, 9)	$t_{HD:DAT}$	Fast mode	0		0.9	µs
		Standard mode	0		0.9	
Data Setup Time (Note 10)	$t_{SU:DAT}$	Fast mode	100			ns
		Standard mode	250			
START Setup Time	$t_{SU:STA}$	Fast mode	0.6			µs
		Standard mode	4.7			
Rise Time of Both SDA and SCL Signals (Note 11)	t_R	Fast mode	20 + 0.1C_B		300	ns
		Standard mode			1000	
Fall Time of Both SDA and SCL Signals (Note 11)	t_F	Fast mode	20 + 0.1C_B		300	ns
		Standard mode			300	
Setup Time for STOP Condition	$t_{SU:STO}$	Fast mode	0.6			µs
		Standard mode	4.7			
Capacitive Load for Each Bus Line	C_B	(Note 11)			400	pF
Capacitance for SDA, SCL	$C_{I/O}$			10		pF
Pulse Width of Spikes That Must Be Suppressed by the Input Filter	t_{SP}			30		ns
Pushbutton Debounce	PB_{DB}			250		ms
Reset Active Time	t_{RST}			250		ms
Oscillator Stop Flag (OSF) Delay	t_{OSF}	(Note 12)		100		ms
Temperature Conversion Time	t_{CONV}			125	200	ms

Power-Switch Characteristics

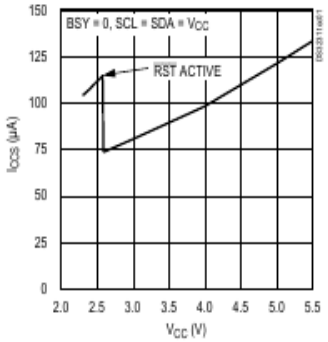
(T_A = T_{MIN} to T_{MAX})

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{CC} Fall Time; $V_{PF(MAX)}$ to $V_{PF(MIN)}$	t_{VCCF}		300			µs
V_{CC} Rise Time; $V_{PF(MIN)}$ to $V_{PF(MAX)}$	t_{VCCR}		0			µs
Recovery at Power-Up	t_{REC}	(Note 13)		250	300	ms

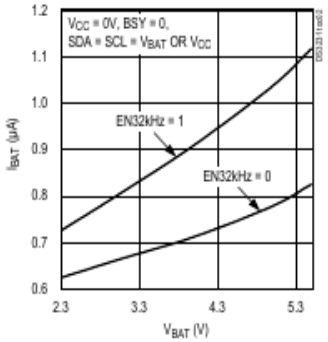
Typical Operating Characteristics

(V_{CC} = +3.3V, T_A = +25°C, unless otherwise noted.)

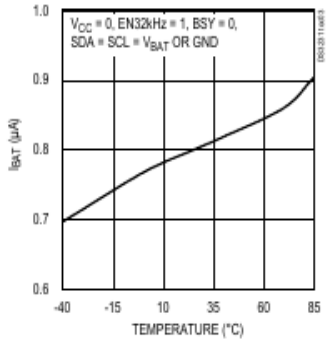
STANDBY SUPPLY CURRENT vs. SUPPLY VOLTAGE



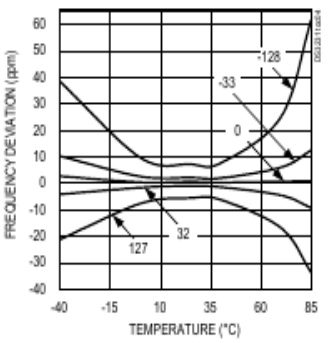
SUPPLY CURRENT vs. SUPPLY VOLTAGE



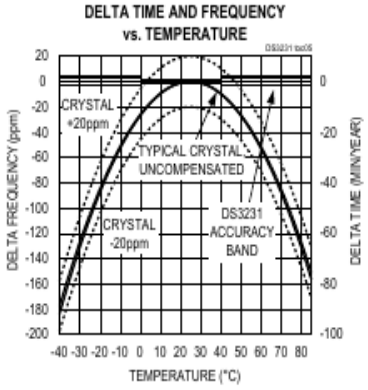
SUPPLY CURRENT vs. TEMPERATURE



FREQUENCY DEVIATION vs. TEMPERATURE vs. AGING VALUE



DELTA TIME AND FREQUENCY vs. TEMPERATURE



Pin Description

PIN	NAME	FUNCTION
1	32kHz	32kHz Output. This open-drain pin requires an external pullup resistor. When enabled, the output operates on either power supply. It may be left open if not used.
2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. If not used, connect to ground.
3	$\overline{INT}$/SQW	Active-Low Interrupt or Square-Wave Output. This open-drain pin requires an external pullup resistor connected to a supply at 5.5V or less. This multifunction pin is determined by the state of the INTCN bit in the Control Register (0Eh). When INTCN is set to logic 0, this pin outputs a square wave and its frequency is determined by RS2 and RS1 bits. When INTCN is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}. If not used, this pin can be left unconnected.
4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is pulled high by the internal pullup resistor. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the oscillator is disabled, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.
5–12	N.C.	No Connection. Must be connected to ground.
13	GND	Ground
14	V_{BAT}	Backup Power-Supply Input. When using the device with the V_{BAT} input as the primary power source, this pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. When using the device with the V_{BAT} input as the backup power source, the capacitor is not required. If V_{BAT} is not used, connect to ground. The device is UL recognized to ensure against reverse charging when used with a primary lithium battery. Go to www.maximintegrated.com/qa/info/ul.
15	SDA	Serial Data Input/Output. This pin is the data input/output for the I2C serial interface. This open-drain pin requires an external pullup resistor. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}.
16	SCL	Serial Clock Input. This pin is the clock input for the I2C serial interface and is used to synchronize data movement on the serial interface. Up to 5.5V can be used for this pin, regardless of the voltage on V_{CC}.

Detailed Description

The DS3231 is a serial RTC driven by a temperature-compensated 32kHz crystal oscillator. The TCXO provides a stable and accurate reference clock, and maintains the RTC to within ±2 minutes per year accuracy from -40°C to +85°C. The TCXO frequency output is available at the 32kHz pin. The RTC is a low-power clock/calendar with two programmable time-of-day alarms and a programmable square-wave output. The $\overline{INT}$/SQW provides either an interrupt signal due to alarm conditions or a square-wave output. The clock/calendar provides 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 $\overline{AM}$/PM indicator. The internal registers are accessible though an I2C bus interface.

A temperature-compensated voltage reference and comparator circuit monitors the level of V_{CC} to detect power failures and to automatically switch to the backup supply when necessary. The $\overline{RST}$ pin provides an external pushbutton function and acts as an indicator of a power-fail event.

Operation

The block diagram shows the main elements of the DS3231. The eight blocks can be grouped into four functional groups: TCXO, power control, pushbutton function, and RTC. Their operations are described separately in the following sections.

Anexo C. Códigos de la aplicación stand-alone

Mainade7880.java

```
package com.tesis2020.medidorade7880;

import java.io.IOException;
import io.dvlopt.linux.i2c.*;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author Felix Campos
 */
public class Mainade7880 {

//REGISTROS DE CONFIGURACION DEL MEDIDOR ADE7880
    public static final byte ADE7880_ADDR_I2C = (byte)0x38;
//Registro RUN para activar la DPS del ADE7880
    public static final byte REG_RUNH = (byte)0xE2;
    public static final byte REG_RUNL = (byte)0x28;
    public static final byte REG_COMPMODEH = (byte)0xE6;
    public static final byte REG_COMPMODEL = (byte)0x0E;
    public static final byte REG_CFMODEH = (byte)0xE6;
    public static final byte REG_CFMODEL = (byte)0x10;
    public static final byte REG_CONFIGH = (byte)0xE6;
    public static final byte REG_CONFIGL = (byte)0x18;
    public static final byte REG_CONFIG2H = (byte)0xEC;
    public static final byte REG_CONFIG2L = (byte)0x01;
    public static final byte REG_CONFIG3H = (byte)0xEA;
    public static final byte REG_CONFIG3L = (byte)0x00;
    public static final byte REG_HCONFIGH = (byte)0xE9;
    public static final byte REG_HCONFIGL = (byte)0x00;
    public static final byte REG_MMODEH = (byte)0xE7;
    public static final byte REG_MMODEL = (byte)0x00;
    public static final byte REG_DSP1H = (byte)0xE7;
    public static final byte REG_DSP1L = (byte)0xFE;
    public static final byte REG_DSP2H = (byte)0xE7;
    public static final byte REG_DSP2L = (byte)0xE2;
    public static final byte REG_VLEVELH = (byte)0x43;
    public static final byte REG_VLEVELL = (byte)0x9F;
    public static final byte REG_CF1DENH = (byte)0xE6;
    public static final byte REG_CF1DENL = (byte)0x11;
    public static final byte DATO_VLEVELH1 = (byte)0x00;
    public static final byte DATO_VLEVELH2 = (byte)0x7A;
    public static final byte DATO_VLEVELL1 = (byte)0x12;
    public static final byte DATO_VLEVELL2 = (byte)0x00;
    public static final byte DATO_CFMODEH1 = (byte)0x0C;
    public static final byte DATO_CFMODEL1 = (byte)0xA0;
    public static final byte DATO_CF1DENH1 = (byte)0x08;
    public static final byte DATO_CF1DENL1 = (byte)0x74;
    public static final byte DATO_DSP1 = (byte)0xAD;
    public static final byte HAB_RAM_DSPH = (byte)0xE7;
    public static final byte HAB_RAM_DSPL = (byte)0xE3;
    public static final byte DATO_HAB_RAM_DSP = (byte)0x80;
    public static final byte DATO_DES_RAM_DSP = (byte)0x00;
    public static final byte DATO_COMPMODEH = (byte)0x41;
    public static final byte DATO_COMPMODEL = (byte)0xFF;
    public static final byte DATO_HCONFIGH = (byte)0x00;
    public static final byte DATO_HCONFIGL_B = (byte)0x0A;
    public static final byte DATO_HCONFIGL_C = (byte)0x0C;
    public static final byte DATO_HCONFIGL_N = (byte)0x0E;
    public static final byte DATO_RUNH = (byte)0x00;
    public static final byte DATO_RUNL = (byte)0x01;
    public static final byte DATO_CONFIG2 = (byte)0x02;
```

```
public static void HabRamDsp() {

            I2CBuffer Dir_Hab1 = new I2CBuffer( 3 ).set( 0, REG_DSP1H )
                                                   .set( 1, REG_DSP1L )
                                                   .set( 2, DATO_DSP1 );
            I2CBuffer Dir_Hab2 = new I2CBuffer( 3 ).set( 0, HAB_RAM_DSPH )
                                                   .set( 1, HAB_RAM_DSPL )
                                                   .set( 2, DATO_HAB_RAM_DSP );
     try {
            I2CBus  i2c = new I2CBus("/dev/i2c-1");
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab1 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            System.out.println( "\t\t" + "Echo HAB PROT RAM DSP" + "\n");
            i2c.close();
         }
            catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void DesRamDsp() {

            I2CBuffer Dir_Hab1 = new I2CBuffer( 3 ).set( 0, REG_DSP1H )
                                                   .set( 1, REG_DSP1L )
                                                   .set( 2, DATO_DSP1 );
            I2CBuffer Dir_Hab2 = new I2CBuffer( 3 ).set( 0, HAB_RAM_DSPH )
                                                   .set( 1, HAB_RAM_DSPL )
                                                   .set( 2, DATO_DES_RAM_DSP );
      try {
            I2CBus  i2c = new I2CBus("/dev/i2c-1");

            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab1 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( Dir_Hab2 );
            System.out.println( "\t\t" + "Echo DES PROT RAM DSP" + "\n");
            i2c.close();
      }
      catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```
public static void initDSPADE7880() {

     2CBuffer DirCOMPMODE_W = new I2CBuffer( 4 ).set( 0, REG_COMPMODEH ).set( 1, REG_COMPMODEL )
                                                .set( 2, DATO_COMPMODEH ).set( 3, DATO_COMPMODEL );
     I2CBuffer DirCONFIG2_W = new I2CBuffer( 3 ).set( 0, REG_CONFIG2H ).set( 1, REG_CONFIG2L )
                                                .set( 2, DATO_CONFIG2 );
     I2CBuffer DIrVLEVEL_W = new I2CBuffer( 6 ).set( 0, REG_VLEVELH).set( 1, REG_VLEVELL)
                                               .set( 2, DATO_VLEVELH1).set( 3, DATO_VLEVELH2)
                                               .set( 4, DATO_VLEVELL1).set( 5, DATO_VLEVELL2);
     I2CBuffer DirCFMODE_W = new I2CBuffer( 4 ).set( 0, REG_CFMODEH ).set( 1, REG_CFMODEL )
                                               .set( 2, DATO_CFMODEH1 ).set( 3, DATO_CFMODEL1 );
     I2CBuffer DirCF1DEN_W = new I2CBuffer( 4 ).set( 0, REG_CF1DENH ).set( 1, REG_CF1DENL )
                                               .set( 2, DATO_CF1DENH1 ).set( 3, DATO_CF1DENL1 );
//   I2CBuffer DirHconfig = new I2CBuffer( 4 ).set( 0, REG_HCONFIGH ).set( 1, REG_HCONFIGL )
//                                             .set( 2, DATO_HCONFIGH ).set( 3, DATO_HCONFIGL_B );
     I2CBuffer DirDSP_W = new I2CBuffer( 4 ).set( 0, REG_RUNH ).set( 1, REG_RUNL )
                                            .set( 2, DATO_RUNH ).set( 3, DATO_RUNL );
      try {
            I2CBus  i2c = new I2CBus("/dev/i2c-1");

            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirCONFIG2_W );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirCOMPMODE_W);
            i2c.selectSlave( ADE7880_ADDR_I2C );
            i2c.write( DIrVLEVEL_W );
            i2c.selectSlave( ADE7880_ADDR_I2C );
            i2c.write( DirCFMODE_W );
            i2c.selectSlave( ADE7880_ADDR_I2C );
            i2c.write( DirCF1DEN_W );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirDSP_W );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirDSP_W );
            i2c.selectSlave( ADE7880_ADDR_I2C );
            i2c.write( DirDSP_W );
            System.out.println( "\t\t" + "Iniciando la DSP del ADE7880" + "\n");
            i2c.close();
      }
      catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("\t\tTrabajo de Graduacion 2020 Ingenieria Electrica\n");

    DesRamDsp();
    initDSPADE7880();
    HabRamDsp();

    Autenticacion sesion = new Autenticacion();
    sesion.setVisible(true);
    }
}
```

Autenticacion.java

```
package com.tesis2020.medidorade7880;

import java.util.List;
import javax.swing.JOptionPane;

/**
 * @author Felix Campos
 */
public class Autenticacion extends javax.swing.JFrame {

    public Autenticacion() {
        initComponents();
    }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        UsuariosJpaController Cusuario = new UsuariosJpaController();
        List<Usuarios> usu;
        usu = Cusuario.findUsuariosEntities();
        String comparausu = jTextusuario.getText();
        String comparacla = jPasswordusuario.getText();

            int usua = 1, k = 0;
            while( usua != 0 ){//comprueba que el usuario existe
                if( ( k + 1 )  == ( usu.size() + 1 ) ) {
                  JOptionPane.showInternalMessageDialog(null, "usuario / contraseña incorrectos");
                }
                if (comparausu.equals(usu.get(k).getUsuario()) &&
comparacla.equals(usu.get(k).getClave())){
                    usua = 0;
                }
                  k++;
            }
            //posicionarme en el ususario y comprobar la bandera
            if( usu.get(k - 1).getBandera() == 1 ){
                OpcionesAdmin pantalla2 = new OpcionesAdmin();
                pantalla2.setVisible(true);
                dispose();
            }
            if( usu.get(k - 1).getBandera() == 0 ) {
                long PME2 = 300000;//5min en milisegundos
                SegundoPlano back2 = new SegundoPlano( PME2 );
                back2.execute();
                System.out.println("\n\t ejecucion en segundo plano iniciada");
                MuestraDatos pantalla6 = new MuestraDatos();
                pantalla6.setVisible(true);
                dispose();
            }
    }private void jTextusuarioKeyTyped(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        if (jTextusuario.getText().length() >= 20) {

            evt.consume();
        }
    }
    private void jPasswordusuarioKeyTyped(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        char validartxt = evt.getKeyChar();
        if ((jPasswordusuario.getText().length() >= 20) && (Character.isLetterOrDigit(validartxt)))
        {
            getToolkit().beep();
            evt.consume();
        }
    }
```

MuestraDatos.java

```
package com.tesis2020.medidorade7880;

import com.opencsv.CSVWriter;
import java.awt.BasicStroke;
import java.awt.Color;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import org.jfree.chart.*;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.*;
import java.awt.image.*;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.plot.XYPlot;
/**
 * @author Felix Campos
 */
public class MuestraDatos extends javax.swing.JFrame {

public static final MedicionescompletasJpaController Cmed = new MedicionescompletasJpaController();
    BufferedImage grafico = null;
    private static int cuent = 0;

    public MuestraDatos() {
        initComponents();
        crearModelo();
    }
     DefaultTableModel modelo;
     private void crearModelo(){
         try {
              modelo = (new javax.swing.table.DefaultTableModel(
             new Object [][] {
                 {null, null, null, null, null, null, null, null, null, null, null, null, null,
                  null, null, null, null, null, null, null,
                  null, null, null, null, null, null, null},
             },
             new String [] {
                 "Registros", "Estampa", "IA", "IB", "IC", "IN", "VA", "VB", "VC", "PActiva A",
                 "PActiva B", "PActiva C", "PReactiva A", "PReactiva B", "PReactiva C",
                 "PAparente A", "PAparente B", "PAparente C",
                 "FP A", "FP B", "FP C", "THDI A", "THDI B", "THDI C", "THDV A", "THDV B", "THDV C"
             }
         ) {
             boolean[] canEdit = new boolean [] {
                 false, false, false, false, false, false, false, false, false, false, false,
                 false, false, false, false, false, false, false, false, false,
                 false, false, false, false, false, false, false
             };
                 public boolean isCellEditable(int rowIndex, int columnIndex) {
                 return canEdit [columnIndex];
             }
         });
               JTableHeader jtableh =  jTabladatos.getTableHeader();
               jtableh.setDefaultRenderer(  new EncabezadoTabla() );
               jTabladatos.setTableHeader(jtableh);
               jTabladatos.setModel(modelo);
          } catch ( Exception e ){
              JOptionPane.showMessageDialog(null, "Error al ejecutar !!!");
          }
     }
```

```
private void exportarCSV() throws IOException, InterruptedException{
    //se crea un objeto lista con los registros de la clase entidad medicionescomplestas
    List<Medicionescompletas> listmed = Cmed.findMedicionescompletasEntities();
    CSVWriter writer = new CSVWriter(new FileWriter("Medicionescompletas.csv"));
    String [] encabezados = { "Registros", "Estampa", "IA", "IB", "IC", "IN", "VA", "VB", "VC",
                              "PActiva A", "PActiva B", "PActiva C", "PReactiva A", "PReactiva B",
                              "PReactiva C", "PAparente A", "PAparente B", "PAparente C","FP A",
                      "FP B", "FP C", "THDI A", "THDI B", "THDI C", "THDV A", "THDV B", "THDV C" };
      writer.writeNext(encabezados);//se escriben los encabezados
      for( int i = 0; i < listmed.size(); i++){//se escriben todos los registros en el archivo csv
           String[] csvBD = { listmed.get(i).getId().toString(), listmed.get(i).getEstampaT(),
                              String.valueOf(listmed.get(i).getIa()),
                              String.valueOf(listmed.get(i).getIb()),
                              String.valueOf(listmed.get(i).getIc()),
                              String.valueOf(listmed.get(i).getIN() ),
                              String.valueOf(listmed.get(i).getVa() ),
                              String.valueOf(listmed.get(i).getVb() ),
                              String.valueOf(listmed.get(i).getVc() ),
                              String.valueOf(listmed.get(i).getPActivaA() ),
                              String.valueOf(listmed.get(i).getPActivaB() ),
                              String.valueOf(listmed.get(i).getPActivaC() ),
                              String.valueOf(listmed.get(i).getPReactivaA() ),
                              String.valueOf(listmed.get(i).getPReactivaB() ),
                              String.valueOf(listmed.get(i).getPReactivaC() ),
                              String.valueOf(listmed.get(i).getPAparenteA() ),
                              String.valueOf(listmed.get(i).getPAparenteB() ),
                              String.valueOf(listmed.get(i).getPAparenteC() ),
                              String.valueOf(listmed.get(i).getFpA() ),
                              String.valueOf(listmed.get(i).getFpB() ),
                              String.valueOf(listmed.get(i).getFpC() ),
                              String.valueOf(listmed.get(i).getThdiA() ),
                              String.valueOf(listmed.get(i).getThdiB() ),
                              String.valueOf(listmed.get(i).getThdiC() ),
                              String.valueOf(listmed.get(i).getThdvA() ),
                              String.valueOf(listmed.get(i).getThdvB() ),
                              String.valueOf(listmed.get(i).getThdvC() )};
                              writer.writeNext(csvBD);
        }
          writer.close();//se cierra el archivo
          JOptionPane.showInternalMessageDialog(null, "Archivo Exportado Con Exito");
    }
private void jBotoncargaActionPerformed(java.awt.event.ActionEvent evt) {
//de desarrolla el metodo para leer los registros de la base de datos y mostrarlos en modelo tabla
        Object objeto[]= null;
        List<Medicionescompletas> listmed = Cmed.findMedicionescompletasEntities();
        int i;

           for(  i = cuent; i < listmed.size(); i++ ){
               modelo.addRow(objeto);
               modelo.setValueAt( listmed.get(i).getId(), i, 0);
               modelo.setValueAt( listmed.get(i).getEstampaT(), i, 1);
               modelo.setValueAt( listmed.get(i).getIa(), i, 2);
               modelo.setValueAt( listmed.get(i).getIb(), i, 3);
               modelo.setValueAt( listmed.get(i).getIc(), i, 4);
               modelo.setValueAt( listmed.get(i).getIN(), i, 5);
               modelo.setValueAt( listmed.get(i).getVa(), i, 6);
               modelo.setValueAt( listmed.get(i).getVb(), i, 7);
               modelo.setValueAt( listmed.get(i).getVc(), i, 8);
               modelo.setValueAt( listmed.get(i).getPActivaA() , i, 9);
               modelo.setValueAt( listmed.get(i).getPActivaB() , i, 10);
               modelo.setValueAt( listmed.get(i).getPActivaC() , i, 11);
               modelo.setValueAt( listmed.get(i).getPReactivaA(), i, 12);
               modelo.setValueAt( listmed.get(i).getPReactivaB(), i, 13);
               modelo.setValueAt( listmed.get(i).getPReactivaC(), i, 14);
               modelo.setValueAt( listmed.get(i).getPAparenteA(), i, 15);
               modelo.setValueAt( listmed.get(i).getPAparenteB(), i, 16);
               modelo.setValueAt( listmed.get(i).getPAparenteC(), i, 17);
               modelo.setValueAt( listmed.get(i).getFpA(), i, 18);
               modelo.setValueAt( listmed.get(i).getFpB(), i, 19);
               modelo.setValueAt( listmed.get(i).getFpC(), i, 20);
               modelo.setValueAt( listmed.get(i).getThdiA(), i, 21);
               modelo.setValueAt( listmed.get(i).getThdiB(), i, 22);
               modelo.setValueAt( listmed.get(i).getThdiC(), i, 23);
               modelo.setValueAt( listmed.get(i).getThdvA(), i, 24);
               modelo.setValueAt( listmed.get(i).getThdvB(), i, 25);
               modelo.setValueAt( listmed.get(i).getThdvC(), i, 26);
           }
           cuent = i;//variable para actualizar el modelo de tabla despues de la primera lectura
    }
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        //se llama el metodo exportarCSV
        try {
            // TODO add your handling code here:
            exportarCSV();
        }
        catch (IOException ex) {
            Logger.getLogger(MuestraDatos.class.getName()).log(Level.SEVERE, null, ex);
        }
        catch (InterruptedException ex) {
            Logger.getLogger(MuestraDatos.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    //metodo para generar las graficas de las corrientes de fase
      List<Medicionescompletas> listmed = Cmed.findMedicionescompletasEntities();
       XYSeries SerieDatos_A = new XYSeries("Grafica de corriente fase A");
       XYSeries SerieDatos_B = new XYSeries("Grafica de corriente fase B");
       XYSeries SerieDatos_C = new XYSeries("Grafica de corriente fase C");

         for( int i = 0; i < listmed.size(); i++ ){

            SerieDatos_A.add( (double) listmed.get(i).getId(), (double) listmed.get(i).getIa() );
            SerieDatos_B.add( (double) listmed.get(i).getId(), (double) listmed.get(i).getIb() );
            SerieDatos_C.add( (double) listmed.get(i).getId(), (double) listmed.get(i).getIc() );

             }
             XYSeriesCollection Datos = new XYSeriesCollection( );
             Datos.addSeries(SerieDatos_A);
             Datos.addSeries(SerieDatos_B);
             Datos.addSeries(SerieDatos_C);

            JFreeChart chart = ChartFactory.createXYLineChart("GRAFICO DE CORRIENTE FASES A, B, C",
                                                              "MUESTRAS", "CORRIENTES RMS [Amperios]",
             Datos, PlotOrientation.VERTICAL, true, true, false);
             XYPlot  plot = chart.getXYPlot();
             NumberAxis  ejeX = (NumberAxis)plot.getDomainAxis();
             NumberAxis  ejeY = (NumberAxis)plot.getRangeAxis();
             ejeX.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
             ejeX.setTickUnit(new NumberTickUnit( 5 ));
             ejeY.setStandardTickUnits( NumberAxis.createIntegerTickUnits());
             ejeY.setTickUnit( new NumberTickUnit(10));
             ejeY.setRange( 0, 110);

             plot.setBackgroundPaint( Color.BLACK );
             plot.setDomainGridlinePaint( Color.WHITE );

             ChartPanel cp = new ChartPanel( chart );
             cp.getChart().getXYPlot().getRenderer().setSeriesPaint( 0, Color.RED );
             cp.getChart().getXYPlot().getRenderer().setSeriesPaint( 1, Color.GREEN );
             cp.getChart().getXYPlot().getRenderer().setSeriesPaint( 2, Color.YELLOW );
             cp.getChart().getXYPlot().getRenderer().setSeriesStroke( 0, new BasicStroke(2.0f));
             cp.getChart().getXYPlot().getRenderer().setSeriesStroke( 1, new BasicStroke(2.0f));
             cp.getChart().getXYPlot().getRenderer().setSeriesStroke( 2, new BasicStroke(2.0f));

             JFrame  ventana = new JFrame("GRAFICOS DE PARAMETROS ELECTRICOS ");
             ventana.setVisible( true );
             ventana.setSize( 800, 600 );
             ventana.add( cp );
    }
```

OpcionesAdmin.java

```
package com.tesis2020.medidorade7880;

/**
 *
 * @author Felix Campos
 */
public class OpcionesAdmin extends javax.swing.JFrame {

    public OpcionesAdmin() {
        initComponents();
    }
private void jButtonremotaActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        ConexionRemota pantalla7 = new ConexionRemota();
        pantalla7.setVisible(true);
        this.setVisible(false);
    }

    private void jButtoncrearMouseClicked(java.awt.event.MouseEvent evt) {

        // TODO add your handling code here:
        NuevoUsuario pantalla3 = new NuevoUsuario();
                pantalla3.setVisible(true);
                this.setVisible(false);
    }

    private void jButtonconfigActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
         ConfigEqmed pantalla6 = new ConfigEqmed();
         pantalla6.setVisible(true);
         this.setVisible(false);
    }

    private void jButtonmuestraActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        MuestraDatos pantalla5 = new MuestraDatos();
        pantalla5.setVisible(true);
        this.setVisible(false);
    }
```

NuevoUsuario.java

```
package com.tesis2020.medidorade7880;

import com.tesis2020.medidorade7880.exceptions.NonexistentEntityException;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 * @author Felix Campos
 */
public class NuevoUsuario extends javax.swing.JFrame {

    UsuariosJpaController Cusuario = new UsuariosJpaController();
    private static final SimpleDateFormat  formato = new SimpleDateFormat( "yyyy-MM-dd HH:mm:ss");

    public NuevoUsuario() {
        initComponents();
      //numero de usuarios anteriores
       jTextFieldid.setText(String.valueOf( Cusuario.getUsuariosCount() ));
    }
 private void jButtonCreaUsuActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
     Usuarios insertar = new Usuarios();
     List<Usuarios> usu = Cusuario.findUsuariosEntities();
     Timestamp  timestm = new Timestamp( System.currentTimeMillis());
     String estampa = String.valueOf(formato.format(timestm));
     String comparausu = jTextFieldusuario.getText();
     String comparacla = jTextFieldclave.getText();
     int k = 1;

     for( int i = 0; i < usu.size(); i++ ){
      if (comparausu.equals(usu.get(i).getUsuario()) && comparacla.equals(usu.get(i).getClave())){
           JOptionPane.showMessageDialog(null, "El usuario ya existe !!!");
       }
       if( k == usu.size() ){
        insertar.setId(Cusuario.getUsuariosCount() + 1 );
        insertar.setEstampa( estampa );
        insertar.setUsuario(jTextFieldusuario.getText());
        insertar.setClave(jTextFieldclave.getText());
        insertar.setBandera(Integer.parseInt(jTextFieldbandera.getText()));
        Cusuario.create( insertar );
          //numero de usuarios actual
        jTextFieldid.setText(String.valueOf(Cusuario.getUsuariosCount() ));
        JOptionPane.showMessageDialog(null, "Usuario creado con exito !!!");
       }
       k++;
      }
    }
  private void jButtonRegreMActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
                OpcionesAdmin pantalla4 = new OpcionesAdmin();
               pantalla4.setVisible(true);
               this.setVisible(false);
    }
```

ConfigEqmed.java

```
package com.tesis2020.medidorade7880;

import static com.tesis2020.medidorade7880.Mainade7880.ADE7880_ADDR_I2C;
import io.dvlopt.linux.gpio.*;
import io.dvlopt.linux.i2c.*;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author Felix Campos
 */
public class ConfigEqmed extends javax.swing.JFrame {

    public static long PME;
    public static final int PIN_ENCENDIDO     = 17 ;
    public static final int PIN_CONEXION_E    = 27 ;
    public static final int PIN_CONEXION_D    = 22 ;
    public static final String GPIO_DEVICE    = "/dev/gpiochip0";

    public ConfigEqmed() {
        initComponents();
    }
  private static void indicadores( GpioHandle handle,
                                   GpioBuffer buffergpio,
                                   GpioLine controlpin,
                                   boolean salidaPin ) {
    try {
         buffergpio.set(controlpin, salidaPin);
         handle.write(buffergpio);
    }
    catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
     }

    private static void encendido( boolean TF ){
        try {
            GpioHandleRequest reques = new GpioHandleRequest().setConsumer("gpioade7880")
                                       .setFlags(new GpioFlags().setOutput());

            GpioLine encendido = reques.addLine( PIN_ENCENDIDO, true );
            GpioBuffer buffergpio = new GpioBuffer();
            GpioDevice device_gpio = new GpioDevice( GPIO_DEVICE );
            GpioHandle handle = device_gpio.requestHandle(reques);
            indicadores( handle, buffergpio, encendido, TF );
            device_gpio.close();
            handle.close();
            System.out.println("\n\t" + "indicaror activado:" + encendido + "\n");
    }
    catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
  private static void estrella( boolean TF ){

     try  {
        GpioHandleRequest reques_Y = new GpioHandleRequest().setConsumer("gpioade7880")
                                     .setFlags(new GpioFlags().setOutput());
            GpioLine conexion_Y = reques_Y.addLine( PIN_CONEXION_E, true );
            GpioBuffer buffergpio_Y = new GpioBuffer();
            GpioDevice device_gpio_Y = new GpioDevice( GPIO_DEVICE );
            GpioHandle handle_Y = device_gpio_Y.requestHandle(reques_Y);
            indicadores( handle_Y, buffergpio_Y, conexion_Y, TF );
            device_gpio_Y.close();
            handle_Y.close();
            System.out.println("\n\t" + "indicador seleccionado:" + conexion_Y + "\n");
     }
     catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```
private static void delta( boolean TF ){

     try  {
        GpioHandleRequest reques_A = new GpioHandleRequest().setConsumer("gpioade7880")
                                          .setFlags(new GpioFlags().setOutput());
            GpioLine conexion_A = reques_A.addLine( PIN_CONEXION_D, true );
            GpioBuffer buffergpio_A = new GpioBuffer();
            GpioDevice device_gpio_A = new GpioDevice( GPIO_DEVICE );
            GpioHandle handle_A = device_gpio_A.requestHandle(reques_A);
            indicadores( handle_A, buffergpio_A, conexion_A, TF );
            device_gpio_A.close();
            handle_A.close();
            System.out.println("\n\t" + "indicador seleccionado:" + conexion_A + "\n");
     }
     catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void jButtonRegreMActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        OpcionesAdmin pantalla4 = new OpcionesAdmin();
        pantalla4.setVisible(true);
        this.setVisible(false);
    }

    private void jCheckI2CActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        try {

        if( jCheckI2C.isSelected()){
           System.out.print("\n\tProtocolo de comunicacion I2C seleccionado\n");
          }
        }
        catch( Exception ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void jButtonIniMedicionesActionPerformed(java.awt.event.ActionEvent evt) {

      // TODO add your handling code here:
        SegundoPlano back = new SegundoPlano( PME );
        back.execute();
        System.out.println("inicio el proceso en segundo plano");
    }
private void jCheckCdeltaActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        try {
        if( jCheckCdelta.isSelected()){

            Mainade7880 initM = new Mainade7880();
            initM.DesRamDsp();

            byte REG_ACCMODEH = (byte)0xE7;
            byte REG_ACCMODEL = (byte)0x01;
            byte DATO_ACCMODE = (byte)0x90;
            I2CBuffer DirAccmode = new I2CBuffer( 3 ).set( 0, REG_ACCMODEH )
                                                     .set( 1, REG_ACCMODEL )
                                                     .set( 2, DATO_ACCMODE );
            I2CBuffer DirAccmode_R = new I2CBuffer( 2 ).set( 0, REG_ACCMODEH )
                                                       .set( 1, REG_ACCMODEL );

            I2CBuffer DatoAccmode_R = new I2CBuffer( 1 );

            I2CBus i2c = new I2CBus("/dev/i2c-1");

            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode_R );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.read(DatoAccmode_R);
            i2c.close();

            initM.initDSPADE7880();
            initM.HabRamDsp();
            System.out.print("\n\tSISTEMA TRIFASICO DELTA SELECCIONADO\n");
          }
        }
        catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```
private void jCheckEstrellaActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
       try {
            if( jCheckEstrella.isSelected()){

            jCheckEstrella.setSelected( true );
            Mainade7888 initM = new Mainade7888();
            initM.DesRamDsp();

            byte REG_ACCMODEH = (byte)0xE7;
            byte REG_ACCMODEL = (byte)0x01;
            byte DATO_ACCMODE = (byte)0x80;
            I2CBuffer DirAccmode = new I2CBuffer( 3 ).set( 0, REG_ACCMODEH )
                                                     .set( 1, REG_ACCMODEL )
                                                     .set( 2, DATO_ACCMODE );
            I2CBuffer DirAccmode_R = new I2CBuffer( 2 ).set( 0, REG_ACCMODEH )
                                                       .set( 1, REG_ACCMODEL );

            I2CBuffer DatoAccmode_R = new I2CBuffer( 1 );

            I2CBus i2c = new I2CBus("/dev/i2c-1");

            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.write( DirAccmode_R );
            i2c.selectSlave( ADE7880_ADDR_I2C ) ;
            i2c.read(DatoAccmode_R);
            i2c.close();

            initM.initDSPADE7880();
            initM.HabRamDsp();
            System.out.print("\n\tSISTEMA TRIFASICO ESTRELLA SELECCIONADO\n");
          }
        }
        catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void jCheckPmediActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        try {

        if( jCheckPmedi.isSelected()){
            jCheckPmedi.setSelected( true );
            PME = 300000;//5min en milisegundos
           System.out.print("\n\t toma de lecturas cada 5min\n");
          }
        }
        catch( Exception ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
private void jCheckHindiActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        if( jCheckHindi.isSelected() ){
            encendido( true );

            if( jCheckEstrella.isSelected() ){
                estrella( true );
                jCheckCdelta.enable(false);
            }else{ estrella( false ); }

            if( jCheckCdelta.isSelected() ){
                delta( true );
                jCheckEstrella.enable(false);
            }else{ delta( false ); }

            System.out.println("\n\t indicadores configurados\n");
        }
        else{ encendido( false ); estrella( false ); delta( false ); }
    }
```

ConexionRemota.java

```
package com.tesis2020.medidorade7880;

import io.dvlopt.linux.gpio.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author Felix Campos
 */
public class ConexionRemota extends javax.swing.JFrame {

    public static final int PIN_HABILITAR_CR  = 23 ;
    public static final String GPIO_DEVICE    = "/dev/gpiochip0";

    public ConexionRemota() {
        initComponents();
    }
  private static void indicadores( GpioHandle handle,
                                   GpioBuffer buffergpio,
                                   GpioLine controlpin,
                                   boolean salidaPin ) {
    try {
         buffergpio.set(controlpin, salidaPin);
         handle.write(buffergpio);
    }
    catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
     }

     private static void Cremoto( boolean TF ){

     try  {
            GpioHandleRequest reques_CR = new GpioHandleRequest().setConsumer("gpioade7880")
                                          .setFlags(new GpioFlags().setOutput());
            GpioLine conexion_CR = reques_CR.addLine( PIN_HABILITAR_CR, true );
            GpioBuffer buffergpio_CR = new GpioBuffer();
            GpioDevice device_gpio_CR = new GpioDevice( GPIO_DEVICE );
            GpioHandle handle_CR = device_gpio_CR.requestHandle(reques_CR);
            indicadores( handle_CR, buffergpio_CR, conexion_CR, TF );
            device_gpio_CR.close();
            handle_CR.close();
            System.out.println("\n\t" + "indicador seleccionado:" + conexion_CR + "\n");
     }
     catch( IOException ex){
        Logger.getLogger(ConfigEqmed.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void RegresaMActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        OpcionesAdmin pantalla4 = new OpcionesAdmin();
        pantalla4.setVisible(true);
        this.setVisible(false);
    }

    private void jCheckHremActionPerformed(java.awt.event.ActionEvent evt) {

        // metodo para habilitar y deshabilitar la conexion remota
        try {
          if( jCheckHrem.isSelected() ){
              Process process = Runtime.getRuntime().exec("sudo systemctl start tomcat");
  BufferedReader buffer = new BufferedReader( new InputStreamReader(process.getInputStream()));
              String resultado = null;
              while( (resultado = buffer.readLine()) != null ){
                  System.out.println( resultado );
              }
              Cremoto( true );
              System.out.println("\n\t iniciando el servidor tomcat");
              System.out.println("\n\t" + buffer );
              }else{ Process process = Runtime.getRuntime().exec("sudo systemctl stop tomcat");
                     Cremoto( false ); }
            }
        catch( IOException ex){
        Logger.getLogger(ConexionRemota.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

EncabezadoTabla.java

```
package com.tesis2020.medidorade7880;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.SwingConstants;
import javax.swing.table.TableCellRenderer;

/**
 * @author Felix Campos
 */
public class EncabezadoTabla implements TableCellRenderer {

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value,
                        boolean isSelected, boolean hasFocus, int row, int column) {

        JComponent jcomponent = null;

        if( value instanceof String ) {
            jcomponent = new JLabel((String) value);
            ((JLabel)jcomponent).setHorizontalAlignment( SwingConstants.CENTER );
            ((JLabel)jcomponent).setSize( 30, jcomponent.getWidth() );
            ((JLabel)jcomponent).setPreferredSize( new Dimension(10, jcomponent.getWidth())  );
        }
        jcomponent.setBorder(javax.swing.BorderFactory.createMatteBorder(0, 0, 1, 1,
                                                new java.awt.Color(255, 255, 255)));
        jcomponent.setOpaque(true);
        jcomponent.setBackground( new Color(65,65,65) );
        jcomponent.setToolTipText("Tabla Seguimiento");
        jcomponent.setForeground(Color.white);

        return jcomponent;
    }
}
```

SegundoPlano.java

```
package com.tesis2020.medidorade7880;


import java.io.IOException;
import io.dvlopt.linux.i2c.*;
import static java.lang.Math.abs;
import static java.lang.Math.pow;
import static java.lang.Math.sqrt;
import static java.lang.Math.acos;
import static java.lang.Math.sin;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.SwingWorker;

/**
 * @author Felix Campos
 */
public class SegundoPlano extends SwingWorker<Void, Void> {

    public static final MedicionescompletasJpaController Medidor = new
MedicionescompletasJpaController();
    private static final SimpleDateFormat  formato = new SimpleDateFormat( "yyyy-MM-dd HH:mm:ss");
    private static final Medicionescompletas datos = new Medicionescompletas();
    public static long Ta;

    public SegundoPlano( long PM ){
        this.Ta = PM;
    }

    private static int convertirByte32aINT(I2CBuffer i2cbuffer) {
        return ((i2cbuffer.get(0) & 0xFF) << 24) |
                    ((i2cbuffer.get(1) & 0xFF) << 16) |
                    ((i2cbuffer.get(2) & 0xFF) << 8) |
                    ((i2cbuffer.get(3) & 0xFF));
    }
    private static int convertirByte16aINT(I2CBuffer i2cbuffer) {
        return ((i2cbuffer.get(0) & 0xFF) << 8) |
                    ((i2cbuffer.get(1) & 0xFF));
    }
    @Override
    protected Void doInBackground( ) throws Exception {

     byte ADE7880_ADDR_I2C = (byte)0x38;
     byte REG_AIRMSH = (byte)0x43;
     byte REG_AIRMSL = (byte)0xC0;
     byte REG_AVRMSH = (byte)0x43;
     byte REG_AVRMSL = (byte)0xC1;
     byte REG_BIRMSH = (byte)0x43;
     byte REG_BIRMSL = (byte)0xC2;
     byte REG_BVRMSH = (byte)0x43;
     byte REG_BVRMSL = (byte)0xC3;
     byte REG_CIRMSH = (byte)0x43;
     byte REG_CIRMSL = (byte)0xC4;
     byte REG_CVRMSH = (byte)0x43;
     byte REG_CVRMSL = (byte)0xC5;
     byte REG_NIRMSH = (byte)0x43;
     byte REG_NIRMSL = (byte)0xC6;
     byte REG_FVRMSH = (byte)0xE8;
     byte REG_FVRMSL = (byte)0x80;
     byte REG_FIRMSH = (byte)0xE8;
     byte REG_FIRMSL = (byte)0x81;
     byte REG_AWATTH = (byte)0xE5;
     byte REG_AWATTL = (byte)0x13;
     byte REG_BWATTH = (byte)0xE5;
     byte REG_BWATTL = (byte)0x14;
     byte REG_CWATTH = (byte)0xE5;
     byte REG_CWATTL = (byte)0x15;
     byte REG_AVAH = (byte)0xE5;
     byte REG_AVAL = (byte)0x19;
     byte REG_BVAH = (byte)0xE5;
     byte REG_BVAL = (byte)0x1A;
     byte REG_CVAH = (byte)0xE5;
     byte REG_CVAL = (byte)0x1B;
     byte REG_APFH = (byte)0xE9;
     byte REG_APFL = (byte)0x02;
     byte REG_BPFH = (byte)0xE9;
     byte REG_BPFL = (byte)0x03;
     byte REG_CPFH = (byte)0xE9;
     byte REG_CPFL = (byte)0x04;
     byte REG_APERIODH = (byte)0xE9;
     byte REG_APERIODL = (byte)0x05;
     byte REG_VTHDH = (byte)0xE8;
     byte REG_VTHDL = (byte)0x86;
     byte REG_ITHDH = (byte)0xE8;
     byte REG_ITHDL = (byte)0x87;
```

```
I2CBuffer DirVfaseA_R = new I2CBuffer( 2 ).set( 0, REG_AVRMSH ).set( 1, REG_AVRMSL );
I2CBuffer DirAperiod_R = new I2CBuffer( 2 ).set( 0, REG_APERIODH ).set( 1, REG_APERIODL );
I2CBuffer DirIfaseA_R = new I2CBuffer( 2 ).set( 0, REG_AIRMSH ).set( 1, REG_AIRMSL );
I2CBuffer DirBirms_R = new I2CBuffer( 2 ).set( 0, REG_BIRMSH ).set( 1, REG_BIRMSL );
I2CBuffer DirBvrms_R = new I2CBuffer( 2 ).set( 0, REG_BVRMSH ).set( 1, REG_BVRMSL );
I2CBuffer DirCirms_R = new I2CBuffer( 2 ).set( 0, REG_CIRMSH ).set( 1, REG_CIRMSL );
I2CBuffer DirCvrms_R = new I2CBuffer( 2 ).set( 0, REG_CVRMSH ).set( 1, REG_CVRMSL );
I2CBuffer DirNirms_R = new I2CBuffer( 2 ).set( 0, REG_NIRMSH ).set( 1, REG_NIRMSL );
I2CBuffer DirAPF_R = new I2CBuffer( 2 ).set( 0, REG_APFH ).set( 1, REG_APFL );
I2CBuffer DirBPF_R = new I2CBuffer( 2 ).set( 0, REG_BPFH ).set( 1, REG_BPFL );
I2CBuffer DirCPF_R = new I2CBuffer( 2 ).set( 0, REG_CPFH ).set( 1, REG_CPFL );
I2CBuffer DirAWATT_R = new I2CBuffer( 2 ).set( 0, REG_AWATTH ).set( 1, REG_AWATTL );
I2CBuffer DirBWATT_R = new I2CBuffer( 2 ).set( 0, REG_BWATTH ).set( 1, REG_BWATTL );
I2CBuffer DirCWATT_R = new I2CBuffer( 2 ).set( 0, REG_CWATTH ).set( 1, REG_CWATTL );
I2CBuffer DirAVA_R = new I2CBuffer( 2 ).set( 0, REG_AVAH ).set( 1, REG_AVAL );
I2CBuffer DirBVA_R = new I2CBuffer( 2 ).set( 0, REG_BVAH ).set( 1, REG_BVAL );
I2CBuffer DirCVA_R = new I2CBuffer( 2 ).set( 0, REG_CVAH ).set( 1, REG_CVAL );
I2CBuffer DirVTHD_R = new I2CBuffer( 2 ).set( 0, REG_VTHDH ).set( 1, REG_VTHDL );
I2CBuffer DirITHD_R = new I2CBuffer( 2 ).set( 0, REG_ITHDH ).set( 1, REG_ITHDL );
I2CBuffer DirHXV_R =new I2CBuffer( 2 ).set(0, REG_HXVRMSH).set(1, REG_HXVRMSL);
I2CBuffer DirHYV_R =new I2CBuffer( 2 ).set(0, REG_HYVRMSH).set(1, REG_HYVRMSL);
I2CBuffer DirHZV_R =new I2CBuffer( 2 ).set(0, REG_HZVRMSH).set(1, REG_HZVRMSL);

I2CBuffer DatoR = new I2CBuffer( 4 );
I2CBuffer DatoRB = new I2CBuffer( 1 );
I2CBuffer DatoVFA = new I2CBuffer( 4 );
I2CBuffer DatoIFA = new I2CBuffer( 4 );
I2CBuffer DatoIFB = new I2CBuffer( 4 );
I2CBuffer DatoVFB = new I2CBuffer( 4 );
I2CBuffer DatoIFC = new I2CBuffer( 4 );
I2CBuffer DatoVFC = new I2CBuffer( 4 );
I2CBuffer DatoIFN = new I2CBuffer( 4 );
I2CBuffer DatoAWATT = new I2CBuffer( 4 );
I2CBuffer DatoBWATT = new I2CBuffer( 4 );
I2CBuffer DatoCWATT = new I2CBuffer( 4 );
I2CBuffer DatoAVA = new I2CBuffer( 4 );
I2CBuffer DatoBVA = new I2CBuffer( 4 );
I2CBuffer DatoCVA = new I2CBuffer( 4 );
I2CBuffer DatoAFVAR = new I2CBuffer( 4 );
I2CBuffer DatoVTHD_A = new I2CBuffer( 4 );
I2CBuffer DatoITHD_A = new I2CBuffer( 4 );
I2CBuffer DatoVTHD_B = new I2CBuffer( 4 );
I2CBuffer DatoITHD_B = new I2CBuffer( 4 );
I2CBuffer DatoVTHD_C = new I2CBuffer( 4 );
I2CBuffer DatoITHD_C = new I2CBuffer( 4 );
I2CBuffer DatoAperiod = new I2CBuffer( 2 );
I2CBuffer DatoAPF = new I2CBuffer( 2 );
I2CBuffer DatoBPF = new I2CBuffer( 2 );
I2CBuffer DatoCPF = new I2CBuffer( 2 );
I2CBuffer DatoHXV = new I2CBuffer( 4 );
I2CBuffer DatoHYV = new I2CBuffer( 4 );
I2CBuffer DatoHZV = new I2CBuffer( 4 );
double v_adc = 0.00, vconv = 0.00, voltajeA = 0.00, voltajeB = 0.00, voltajeC = 0.00,
       APF = 0.00, BPF = 0.00, CPF = 0.00, ATHDV = 0.00, ATHDI = 0.00, BTHDV = 0.00,
       BTHDI = 0.00, CTHDV = 0.00, CTHDI = 0.00, iconv = 0.00, corriente = 0.00,
       corrienteB = 0.00, corrienteC = 0.00, corrienteN = 0.00, pot_a = 0.00, pot_b = 0.00,
       pot_c = 0.00, watt_lsb = 0.00, pva_lsb = 0.00, papa_a = 0.00, papa_b = 0.00,
       papa_c = 0.00, pvar_a = 0.00, pvar_b = 0.00, pvar_c = 0.00, hxv = 0.00, hyv = 0.00,
       hzv = 0.00;
       int   datovf = 0, datoif = 0, datoFP = 0, datowatt = 0, datopva = 0, datopvar = 0,
             datothdv = 0, datothdi = 0;
       v_adc = (1.2/sqrt(2)) / pow( 2, 23 );
       int ht = 6 ;
```

```
try {
     I2CBus  i2c = new I2CBus("/dev/i2c-1");

     while( ht != 0 ){

         Thread.sleep( Ta );
         ht--;
     Timestamp  timesth = new Timestamp( System.currentTimeMillis());
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirAperiod_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoAperiod );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirVfaseA_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoVFA );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirIfaseA_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoIFA );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirBirms_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoIFB );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirBvrms_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoVFB );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirCirms_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoIFC );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirCvrms_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoVFC );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirNirms_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoIFN );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirAPF_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoAPF );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirBPF_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoBPF );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirCPF_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoCPF );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirAWATT_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoAWATT );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirBWATT_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoBWATT );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.write( DirCWATT_R );
     i2c.selectSlave( ADE7880_ADDR_I2C );
     i2c.read( DatoCWATT );
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirAVA_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoAVA);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirBVA_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoBVA);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirCVA_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoCVA);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirVTHD_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoVTHD_A);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirITHD_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoITHD_A);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirHXV_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoHXV);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirHYV_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoHYV);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.write(DirHZV_R);
     i2c.selectSlave(ADE7880_ADDR_I2C);
     i2c.read(DatoHZV);
```

```
datovf = convertirByte32aINT( DatoVFA ); //convierte loa 4 bytes en un numero entero
     vconv = v_adc * datovf;
     voltajeA = ( vconv * 600.50);
     System.out.println( "numero de la conversion ADC\t\t" + datovf + "\n");
     System.out.println( "Voltaje convertido ADC\t\t" + vconv + "\n");
     System.out.println( "Voltaje A RMS medido\t\t" + voltajeA + "\n");

     datoif = convertirByte32aINT( DatoIFA );
     iconv = v_adc * datoif;
     corriente = ( iconv * 315.714 );
     System.out.println( "numero de la conversion ADC\t\t" + datoif + "\n");
     System.out.println( "Corriente convertida ADC\t\t" + iconv + "\n");
     System.out.println( "Corriente A RMS medido\t\t" + corriente + "\n");

     datovf = convertirByte32aINT( DatoVFB );
     vconv = v_adc * datovf;
     voltajeB = ( vconv * 600.50);
     System.out.println( "numero de la conversion ADC\t\t" + datovf + "\n");
     System.out.println( "Voltaje convertido ADC\t\t" + vconv + "\n");
     System.out.println( "Voltaje B RMS medido\t\t" + voltajeB + "\n");

     datoif = convertirByte32aINT( DatoIFB );
     iconv = v_adc * datoif;
     corrienteB = ( iconv * 285.714 );
     System.out.println( "numero de la conversion ADC\t\t" + datoif + "\n");
     System.out.println( "Corriente convertida ADC\t\t" + iconv + "\n");
     System.out.println( "Corriente B RMS medido\t\t" + corrienteB + "\n");

     datovf = convertirByte32aINT( DatoVFC );
     vconv = v_adc * datovf;
     voltajeC = ( vconv * 600.50);
     System.out.println( "numero de la conversion ADC\t\t" + datovf + "\n");
     System.out.println( "Voltaje convertido ADC\t\t" + vconv + "\n");
     System.out.println( "Voltaje B RMS medido\t\t" + voltajeB + "\n");

     datoif = convertirByte32aINT( DatoIFC );
     iconv = v_adc * datoif;
     corrienteC = ( iconv * 285.714 ); //285.714
     System.out.println( "numero de la conversion ADC\t\t" + datoif + "\n");
     System.out.println( "Corriente convertida ADC\t\t" + iconv + "\n");
     System.out.println( "Corriente B RMS medido\t\t" + corrienteB + "\n");

     datoif = convertirByte32aINT( DatoIFN );
     iconv = v_adc * datoif;
     corrienteN = ( iconv * 285.714 ); //285.714
     System.out.println( "numero de la conversion ADC\t\t" + datoif + "\n");
     System.out.println( "Corriente convertida ADC\t\t" + iconv + "\n");
     System.out.println( "Corriente B RMS medido\t\t" + corrienteB + "\n");

     datoFP = convertirByte16aINT( DatoAPF );
     System.out.println( "Valor del registro APF en INT\t\t" + datoFP + "\n");
     if( datoFP > 0 && datoFP <= 32767 ){
        APF = datoFP / pow( 2, 15 );}
     if( datoFP > 32768 ){
        APF = ( datoFP / pow( 2, 16 ) * -1);  }
     System.out.println( "Conversion del registro APF\t\t" + datoFP + "\n");
     System.out.println( "Factor de Potencia fase A\t\t" + APF + "\n");

     datoFP = convertirByte16aINT( DatoBPF );
     System.out.println( "Valor del registro BPF en INT\t\t" + datoFP + "\n");
     if( datoFP > 0 && datoFP <= 32767 ){
        BPF = datoFP / pow( 2, 15 ); }
     if( datoFP > 32768 ){
        BPF = datoFP / pow( 2, 16 ) * -1;}
     System.out.println( "Conversion del registro BPF\t\t" + datoFP + "\n");
     System.out.println( "Factor de Potencia fase B\t\t" + BPF + "\n");

     datoFP = convertirByte16aINT( DatoCPF );
     System.out.println( "Valor del registro CPF en INT\t\t" + datoFP + "\n");
     if( datoFP > 0 && datoFP <= 32767 ){
        CPF = datoFP / pow( 2, 15 );}
      if( datoFP > 32768 ){
         CPF = datoFP / pow( 2, 16 ) * -1;}
     System.out.println( "Conversion del registro CPF\t\t" + datoFP + "\n");
     System.out.println( "Factor de Potencia fase C\t\t" + CPF + "\n");
```

```
datowatt = convertirByte32aINT( DatoAWATT );
System.out.println("conversion del registro AWATT\t\t" + datowatt + "\n");
watt_lsb = (voltajeA * corriente * abs(APF)) / datowatt;
pot_a = watt_lsb * datowatt;
System.out.println("Medicion de Potencia fase A en Watt\t\t" + pot_a + "\n");

datowatt = convertirByte32aINT( DatoBWATT );
System.out.println("conversion del registro BWATT\t\t" + datowatt + "\n");
watt_lsb = (voltajeB * corrienteB * abs(BPF)) / datowatt;
pot_b = watt_lsb * datowatt;
System.out.println("Medicion de Potencia fase B en Watt\t\t" + pot_b + "\n");

datowatt = convertirByte32aINT( DatoCWATT );
System.out.println("conversion del registro CWATT\t\t" + datowatt + "\n");
watt_lsb = (voltajeC * corrienteC * abs(CPF)) / datowatt;
pot_c = watt_lsb * datowatt;
System.out.println("Medicion de Potencia fase C en Watt\t\t" + pot_c + "\n");

datopva = convertirByte32aINT( DatoAVA );
System.out.println("conversion del registro AVA\t\t" + datopva + "\n");
pva_lsb = (voltajeA * corriente) / datopva;
papa_a = pva_lsb * datopva;
System.out.println("Medicion de Potencia Aparente fase A\t\t" + papa_a + "\n");


datopva = convertirByte32aINT( DatoBVA );
System.out.println("conversion del registro BVA\t\t" + datopva + "\n");
pva_lsb = (voltajeB * corrienteB) / datopva;
papa_b = pva_lsb * datopva;
System.out.println("Medicion de Potencia Aparente fase B\t\t" + papa_b + "\n");


datopva = convertirByte32aINT( DatoCVA );
System.out.println("conversion del registro BVA\t\t" + datopva + "\n");
pva_lsb = (voltajeC * corrienteC) / datopva;
papa_c = pva_lsb * datopva;
System.out.println("Medicion de Potencia Aparente fase C\t\t" + papa_c + "\n");


pvar_a = (voltajeA * corriente * sin( acos(APF)));
System.out.println("angulo theta A\t\t" + acos(APF) + "\n");
System.out.println("Medicion de Potencia Reactiva fase A\t\t" + pvar_a + "\n");


pvar_b = (voltajeB * corrienteB * sin( acos(BPF)));
System.out.println("angulo theta B\t\t" + acos(BPF) + "\n");
System.out.println("Medicion de Potencia Reactiva fase B\t\t" + pvar_b + "\n");

pvar_c = (voltajeC * corrienteC * sin( acos(CPF)));
System.out.println("angulo theta C\t\t" + acos(CPF) + "\n");
System.out.println("Medicion de Potencia Reactiva fase C\t\t" + pvar_c + "\n");

datothdv = convertirByte32aINT( DatoVTHD_A );
ATHDV = ( datothdv / pow( 2, 21 ) ) * 100;
System.out.println("conversion del registro THDV\t\t" + datothdv + "\n");
System.out.println("THD_V para la fase A\t\t" + ATHDV + "\n");

datothdi = convertirByte32aINT( DatoITHD_A );
ATHDI = ( datothdi / pow( 2, 21 ) ) * 100;
System.out.println("conversion del registro THDI\t\t" + datothdi + "\n");
System.out.println("THD_I para la fase A\t\t" + ATHDI + "\n");

datothdv = convertirByte32aINT( DatoVTHD_B );
BTHDV = ( datothdv / pow( 2, 21 ) ) * 100;
System.out.println("conversion del registro THDV\t\t" + datothdv + "\n");
System.out.println("THD_V para la fase B\t\t" + BTHDV + "\n");

datothdi = convertirByte32aINT( DatoITHD_B );
BTHDI = ( datothdi / pow( 2, 21 ) ) * 100;
System.out.println("conversion del registro THDI\t\t" + datothdi + "\n");
System.out.println("THD_I para la fase B\t\t" + BTHDI + "\n");

hxv = convertirByte32aINT( DatoHXV );
System.out.println("conversion del registro\t\t" + hxv + "\n");

hyv = convertirByte32aINT( DatoHYV );
System.out.println("conversion del registro\t\t" + hyv + "\n");

hzv = convertirByte32aINT( DatoHZV );
System.out.println("conversion del registro\t\t" + hzv + "\n");
```

```
            voltajeA = Math.round( voltajeA * 100 ) / 100d ;
            voltajeB = Math.round( voltajeB * 100 ) / 100d ;
            voltajeC = Math.round( voltajeC * 100 ) / 100d ;
            corriente = Math.round( corriente * 100 ) / 100d ;
            corrienteB = Math.round( corrienteB * 100 ) / 100d ;
            corrienteC = Math.round( corrienteC * 100 ) / 100d ;
            corrienteN = Math.round( corrienteN * 100 ) / 100d ;
            pot_a = Math.round( pot_a * 100 ) / 100d ;
            pot_b = Math.round( pot_b * 100 ) / 100d ;
            pot_c = Math.round( pot_c * 100 ) / 100d ;
            papa_a = Math.round( papa_a * 100 ) / 100d ;
            papa_b = Math.round( papa_b * 100 ) / 100d ;
            papa_c = Math.round( papa_c * 100 ) / 100d ;
            pvar_a = Math.round( pvar_a * 100 ) / 100d ;
            pvar_b = Math.round( pvar_b * 100 ) / 100d ;
            pvar_c = Math.round( pvar_c * 100 ) / 100d ;
            APF = Math.round( APF * 100 ) / 100d ;
            BPF = Math.round( BPF * 100 ) / 100d ;
            CPF = Math.round( CPF * 100 ) / 100d ;
            ATHDV = Math.round( ATHDV * 100 ) / 100d ;
            ATHDI = Math.round( ATHDI * 100 ) / 100d ;
            BTHDV = Math.round( BTHDV * 100 ) / 100d ;
            BTHDI = Math.round( BTHDI * 100 ) / 100d ;

//*************************************************************************//

            String estampa = String.valueOf(formato.format(timesth));
            datos.setId( Medidor.getMedicionescompletasCount() + 1 );
            datos.setEstampaT(estampa);
            datos.setIa( corriente );
            datos.setIb( corrienteB );
            datos.setIc( corrienteC );
            datos.setIN( corrienteN );
            datos.setVa( voltajeA );
            datos.setVb( voltajeB );
            datos.setVc( voltajeC );
            datos.setPActivaA(pot_a);
            datos.setPActivaB(pot_b);
            datos.setPActivaC(pot_c);
            datos.setPReactivaA(pvar_a);
            datos.setPReactivaB(pvar_b);
            datos.setPReactivaC(pvar_c);
            datos.setPAparenteA(papa_a);
            datos.setPAparenteB(papa_b);
            datos.setPAparenteC(papa_c);
            datos.setFpA(APF);
            datos.setFpB(BPF);
            datos.setFpC(CPF);
            datos.setThdiA(ATHDI);
            datos.setThdvA(ATHDV);
            datos.setThdiB(BTHDI);
            datos.setThdvB(BTHDV);
            Medidor.create(datos);
       }
            i2c.close();
    }
    catch (IOException ex) {
            Logger.getLogger(Mainade7880.class.getName()).log(Level.SEVERE, null, ex);
        }
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

Anexo D. Códigos de la aplicación web lado del servidor

Index.jsp

```
<%--
    Document   : index
    Created on : 27 may. 2021, 00:49:36
    Author     : Felix Campos
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title aling="center">SERVIDOR - MEDIDOR TRIFASICO ADE7880</title>
    </head>
    <body>
        <%
            HttpSession ses = request.getSession();
            if( ses.getAttribute("usuario") == null && ses.getAttribute("clave") == null ){
                response.sendRedirect("Autenticacion.jsp");
            }
        %>
        <h1>SISTEMA DE AUTENTICACION DE USUARIOS</h1>
    </body>
</html>
```

Autenticacion.jsp

```
<%--
    Document   : Autenticacion
    Created on :
    Author     : Felix Campos
--%>

<%@page import="java.util.List"%>
<%@page import="com.tesis2020.servidorade7880.Usuarios"%>
<%@page import="com.tesis2020.servidorade7880.UsuariosJpaController"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>SERVIDOR - MEDIDOR TRIFASICO ADE7880</title>
    </head>
    <%
        UsuariosJpaController cusu = new UsuariosJpaController();
        List<Usuarios> lusu = cusu.findUsuariosEntities();
        HttpSession Saut = request.getSession();
        int usua = 1, i = 0;

        if( request.getParameter("btningresar") != null ){
            String usuario = request.getParameter("textusuario").toString();
            String clave = request.getParameter("textclave").toString();

            while( usua != 0 ){
                if( lusu.get(i).getUsuario().equals(usuario) &&
lusu.get(i).getClave().equals(clave)){
//                    out.print("<script>" + "alert('Bienvenido Admin !!!')" + "</script>");
                    if( lusu.get(i).getBandera() == 1){
                    response.sendRedirect("MuestraDatos.jsp");
                    usua = 0;}
                    if( lusu.get(i).getBandera() == 0){
                    response.sendRedirect("MuestraDatos.jsp");
                    usua = 0; }
                }
                 i++;
            } out.print("<script>alert('Usuario / contraseña incorrectos')</script>");
        }
    %>
    <body>
    <center><h1>Bienvenido al sistema de sesiones remotas!!!</h1></center>
    <hr width="60%"/>
    <br/>
    <br/>
    <br/>
    <div class="form-group">
        <center><form action="Autenticacion.jsp" method="POST">
            <input class="form-control" type="text" name="textusuario" placeholder="Usuario"><br>
            <input class="form-control" type="password" name="textclave" placeholder="Password">
<br><br>
            <input  class="btn btn-success" type="submit" name="btningresar" value="Ingresar">
            </form></center>
    </div>
    </body>
</html>
```

MuestraDatos.jsp

```
<%--
    Document   : MuestraDatos
    Created on :
    Author     : Felix Campos
--%>

<%@page import="java.util.List"%>
<%@page import="com.tesis2020.servidorade7880.Medicionescompletas"%>
<%@page import="com.tesis2020.servidorade7880.MedicionescompletasJpaController"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>SERVIDOR - MEDIDOR TRIFASICO ADE7880</title>
    </head>
    <body>
    <center><h1>REGISTROS DE LAS MEDICIONES</h1></center>
    <br/>
    <br/>
    <br/>
    <center><table border="2"></center>
    <tr>
        <td><b>Registros</b></td>
                <td><center><b>Estampa</b></center></td>
                <td><center><b>IA</b></center></td>
                <td><center><b>IB</b></center></td>
                <td><center><b>IC</b></center></td>
                <td><center><b>IN</b></center></td>
                <td><center><b>VA</b></center></td>
                <td><center><b>VB</b></center></td>
                <td><center><b>VC</b></center></td>
                <td><center><b>PActivaA</b></center></td>
                <td><center><b>PActivaB</b></center></td>
                <td><center><b>PActivaC</b></center></td>
                <td><b>PReactivaA</b></td>
                <td><b>PReactivaB</b></td>
                <td><b>PReactivaC</b></td>
                <td><b>PAparenteA</b></td>
                <td><b>PAparenteB</b></td>
                <td><b>PAparenteC</b></td>
                <td><center><b>FPA</b></center></td>
                <td><b>FPB</b></td>
                <td><b>FPC</b></td>
                <td><center><b>THDIA</b></center></td>
                <td><b>THDIB</b></td>
                <td><b>THDIC</b></td>
                <td><b>THDVA</b></td>
                <td><b>THDVB</b></td>
                <td><b>THDVC</b></td>
            </tr>
             <%
                  if( request.getParameter( "btndatos" ) != null ){
        MedicionescompletasJpaController cmed = new MedicionescompletasJpaController();
        List<Medicionescompletas>  lmed = cmed.findMedicionescompletasEntities();
        double cia = 0.00, vva = 0.00;
        for( int i = 0; i < lmed.size(); i++){
        out.println( "<tr>");
        out.println( "<td>" + "<center>" + lmed.get(i).getId() + "</center>" + "</td>");
        out.println( "<td>" + lmed.get(i).getEstampaT() + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getIa() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getIb() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getIc() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getIN() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getVa() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getVb() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getVc() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPActivaA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPActivaB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPActivaC() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPReactivaA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPReactivaB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPReactivaC() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPAparenteA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPAparenteB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getPAparenteC() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getFpA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getFpB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getFpC() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdiA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdiB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdiC() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdvA() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdvB() + "</center>" + "</td>");
        out.println( "<td>" + "<center>" + lmed.get(i).getThdvC() + "</center>" + "</td>");
        out.println( "</tr>");
        }
        out.println( "</table>" );
                  }
    %>
    <center><form action="MuestraDatos.jsp" method="POST">
    <input  type="submit" name="btndatos" value="Cargar Datos">
        </form></center>
    </body>
</html>
```

GraficoCorriente.jsp

```
<%--
    Document   : GraficoCoriente
    Created on :
    Author     : Felix Campos
--%>

<%@page import="javax.swing.Spring"%>
<%@page import="org.jfree.chart.ChartUtilities"%>
<%@page import="java.io.OutputStream"%>
<%@page import="java.awt.BasicStroke"%>
<%@page import="org.jfree.chart.ChartPanel"%>
<%@page import="java.awt.Color"%>
<%@page import="org.jfree.chart.axis.NumberTickUnit"%>
<%@page import="org.jfree.chart.axis.NumberAxis"%>
<%@page import="org.jfree.data.xy.XYSeriesCollection"%>
<%@page import="org.jfree.chart.plot.XYPlot"%>
<%@page import="org.jfree.chart.plot.PlotOrientation"%>
<%@page import="org.jfree.chart.JFreeChart"%>
<%@page import="org.jfree.chart.ChartFactory"%>
<%@page import="org.jfree.data.xy.XYSeries"%>
<%@page import="java.util.List"%>
<%@page import="com.tesis2020.servidorade7880.Medicionescompletas"%>
<%@page import="com.tesis2020.servidorade7880.MedicionescompletasJpaController"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>SERVIDOR - MEDIDOR TRIFASICO ADE7880</title>
    </head>
    <body>
    <center><h1>GRAFICOS DE CORRIENTE</h1></center>
        <%
            if( request.getParameter("btngraficogc") != null ){
            MedicionescompletasJpaController GCmed = new MedicionescompletasJpaController();
            List<Medicionescompletas>  GClist = GCmed.findMedicionescompletasEntities();

            XYSeries SDatosIA = new XYSeries("Grafica de corriente fase A");
            XYSeries SDatosIB = new XYSeries("Grafica de corriente fase B");
            XYSeries SDatosIC = new XYSeries("Grafica de corriente fase C");

            for( int i = 0; i < GClist.size(); i++ ){
               SDatosIA.add( (double) GClist.get(i).getId(), (double) GClist.get(i).getIa());
               SDatosIB.add( (double) GClist.get(i).getId(), (double) GClist.get(i).getIb());
               SDatosIC.add( (double) GClist.get(i).getId(), (double) GClist.get(i).getIc());
            }

            XYSeriesCollection Datos = new XYSeriesCollection( );
             Datos.addSeries(SDatosIA);
             Datos.addSeries(SDatosIB);
             Datos.addSeries(SDatosIC);

            JFreeChart chartgc = ChartFactory.createXYLineChart("GRAFICO DE CORRIENTES DE FASE",
"MUESTRAS", "CORRIENTES RMS [Amperios]", Datos, PlotOrientation.VERTICAL, true, true, false);
            XYPlot  plot = chartgc.getXYPlot();

            NumberAxis  ejeX = (NumberAxis)plot.getDomainAxis();
            NumberAxis  ejeY = (NumberAxis)plot.getRangeAxis();
            ejeX.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
            ejeX.setTickUnit(new NumberTickUnit( 20 ));
            ejeY.setStandardTickUnits( NumberAxis.createIntegerTickUnits());
            ejeY.setTickUnit( new NumberTickUnit(10));
            ejeY.setRange( 0, 110);

            plot.setBackgroundPaint( Color.BLACK );
            plot.setDomainGridlinePaint( Color.WHITE );

            chartgc.getXYPlot().getRenderer().setSeriesPaint(0, Color.RED);
            chartgc.getXYPlot().getRenderer().setSeriesPaint(1, Color.GREEN);
            chartgc.getXYPlot().getRenderer().setSeriesPaint(2, Color.YELLOW);

            response.setContentType("image/JPEG");
            OutputStream ops = response.getOutputStream();
//            ChartUtilities.writeChartAsPNG(ops, chartgc, 640, 480 );//640,480
            ChartUtilities.writeChartAsJPEG(ops, chartgc, 640, 480);
//            out.println( "<td>" + "<img src='GraficoCorriente.jsp'/>" + "</td>" );
            }
        %>
        <center><form action="GraficoCorriente.jsp" method="POST">
        <input  type="submit" name="btngraficogc" value="Graficar Corrientes">
        </form></center>
    </body>
</html>
```

Anexo E. Códigos de la aplicación web lado del cliente

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com
/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/design_default_color_error"
        android:baselineAligned="true"
        android:orientation="vertical"
        android:theme="@style/Theme.EQUIPODEMEDICIONTRIFASICO.AppBarOverlay">

        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:minHeight="?actionBarSize"
            android:padding="@dimen/appbar_padding"
            android:text="@string/app_name"
            android:textAppearance="@style/TextAppearance.Widget.AppCompat.Toolbar.Title" />

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/design_default_color_error"
            android:isScrollContainer="false"
            android:scrollbarAlwaysDrawHorizontalTrack="false"
            app:tabIndicatorColor="@android:color/holo_orange_light"
            app:tabTextColor="@color/black" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_1" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_2" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_3" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_4" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_5" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_6" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_7" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tab_text_8" />

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Fragment_gcorriente.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".GcorrienteFragment">

    <!-- TODO: Update blank fragment layout -->

    <WebView
        android:id="@+id/web_page_gc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</FrameLayout>
```

Strings.xml

```
<resources>
    <string name="app_name">EQUIPO DE MEDICION TRIFASICO</string>
    <string name="tab_text_1">TABLA DE MEDICIONES</string>
    <string name="tab_text_2">GRAFICO DE CORRIENTES</string>
    <string name="tab_text_3">GRAFICO DE VOLTAJES</string>
    <string name="tab_text_4">GRAFICO DE POTENCIAS</string>
    <string name="tab_text_5">GRAFICO DE POTENCIAS REACTIVAS</string>
    <string name="tab_text_6">GRAFICO DE POTENCIAS APARENTES</string>
    <string name="tab_text_7">GRAFICO DE THD DE CORRIENTE</string>
    <string name="tab_text_8">GRAFICO DE THD DE VOLTAJE</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
</resources>
```

Themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.EQUIPODEMEDICIONTRIFASICO"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/rojo</item>
        <item name="colorPrimaryVariant">@color/black</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>

    <style name="Theme.EQUIPODEMEDICIONTRIFASICO.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>

    <style name="Theme.EQUIPODEMEDICIONTRIFASICO.AppBarOverlay"
parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

    <style name="Theme.EQUIPODEMEDICIONTRIFASICO.PopupOverlay"
parent="ThemeOverlay.AppCompat.Light" />
</resources>
```

Themes.xml(night)

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.EQUIPODEMEDICIONTRIFASICO"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/rojo</item>
        <item name="colorPrimaryVariant">@color/rojo</item>
        <item name="colorOnPrimary">@color/black</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Mainactivity.java

```
package com.tesis2020.equipodemediciontrifasico;

import android.content.Context;
import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.tabs.TabLayout;

import androidx.viewpager.widget.ViewPager;
import androidx.appcompat.app.AppCompatActivity;


import android.view.View;


import com.tesis2020.equipodemediciontrifasico.ui.main.SectionsPagerAdapter;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SectionsPagerAdapter sectionsPagerAdapter = new SectionsPagerAdapter(this,
getSupportFragmentManager());
        ViewPager viewPager = findViewById(R.id.view_pager);
        viewPager.setAdapter(sectionsPagerAdapter);
        TabLayout tabs= findViewById(R.id.tabs);
        tabs.setupWithViewPager(viewPager);
        Context context = getApplicationContext();
    }
}
```

PageViewModel.java

```
package com.tesis2020.equipodemediciontrifasico.ui.main;

import androidx.arch.core.util.Function;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Transformations;
import androidx.lifecycle.ViewModel;

public class PageViewModel extends ViewModel {

    private MutableLiveData<Integer> mIndex = new MutableLiveData<>();
    private LiveData<String> mText = Transformations.map(mIndex, new Function<Integer, String>() {
        @Override
        public String apply(Integer input) {
            return " MEDIDOR TRIFASICO: " + input;
        }
    });

    public void setIndex(int index) {
        mIndex.setValue(index);
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

PlaceholderFragment.java

```
package com.tesis2020.equipodemediciontrifasico.ui.main;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.Nullable;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import com.tesis2020.equipodemediciontrifasico.R;


public class PlaceholderFragment extends Fragment {

    private static final String ARG_SECTION_NUMBER = "section_number";

    private PageViewModel pageViewModel;

    public static PlaceholderFragment newInstance(int index) {
        PlaceholderFragment fragment = new PlaceholderFragment();
        Bundle bundle = new Bundle();
        bundle.putInt(ARG_SECTION_NUMBER, index);
        fragment.setArguments(bundle);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        pageViewModel = new ViewModelProvider(this).get(PageViewModel.class);
        int index = 1;
        if (getArguments() != null) {
            index = getArguments().getInt(ARG_SECTION_NUMBER);
        }
        pageViewModel.setIndex(index);
    }

    @Override
    public View onCreateView(
            @NonNull LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_main, container, false);
        final TextView textView = root.findViewById(R.id.section_label);
        pageViewModel.getText().observe(this, new Observer<String>() {
            @Override
            public void onChanged(@Nullable String s) {
                textView.setText(s);
            }
        });
        return root;
    }
}
```

SelectionspageAdapter.java

```
package com.tesis2020.equipodemediciontrifasico.ui.main;

import android.content.Context;

import androidx.annotation.Nullable;
import androidx.annotation.StringRes;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;

import com.tesis2020.equipodemediciontrifasico.GcorrienteFragment;
import com.tesis2020.equipodemediciontrifasico.GpactivaFragment;
import com.tesis2020.equipodemediciontrifasico.GpaparenteFragment;
import com.tesis2020.equipodemediciontrifasico.GpreactivaFragment;
import com.tesis2020.equipodemediciontrifasico.GthdiFragment;
import com.tesis2020.equipodemediciontrifasico.GthdvFragment;
import com.tesis2020.equipodemediciontrifasico.GvoltajeFragment;
import com.tesis2020.equipodemediciontrifasico.MedicionesFragment;
import com.tesis2020.equipodemediciontrifasico.R;

/**
 * Un [FragmentPagerAdapter] que devuelve un fragmento correspondiente a
 * una de las secciones/pestañas/páginas.
 */
public class SectionsPagerAdapter extends FragmentPagerAdapter {

    @StringRes
    private static final int[] TAB_TITLES = new int[]{R.string.tab_text_1, R.string.tab_text_2,
                                                      R.string.tab_text_3, R.string.tab_text_4,
                                                      R.string.tab_text_5, R.string.tab_text_6,
                                                      R.string.tab_text_7, R.string.tab_text_8};
    private final Context mContext;

    public SectionsPagerAdapter(Context context, FragmentManager fm) {
        super(fm);
        mContext = context;
    }

    @Override
    public Fragment getItem(int position) {
//getItem es llamado para instanciar el fragmento de cada página seleccionada.
//Devuelve un PlaceholderFragment (definido como una clase interna estática más abajo).
//return PlaceholderFragment.newInstance(position + 1);
        switch ( position ){
            case 0:
                return new MedicionesFragment();
            case 1:
                return new GcorrienteFragment();
            case 2:
                return new GvoltajeFragment();
            case 3:
                return new GpactivaFragment();
            case 4:
                return new GpreactivaFragment();
            case 5:
                return new GpaparenteFragment();
            case 6:
                return new GthdiFragment();
            case 7:
                return new GthdvFragment();
            default:
                return null;

        }
    }

    @Nullable
    @Override
    public CharSequence getPageTitle(int position) {
        return mContext.getResources().getString(TAB_TITLES[position]);
    }

    @Override
    public int getCount() {
   // muestra todas las paginas.
        return 8;
    }
}
```

GcorrienteFragment.java

```
package com.tesis2020.equipodemediciontrifasico;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.webkit.WebViewClient;


public class GcorrienteFragment extends Fragment {

    public GcorrienteFragment() {
        // Constructor público vacío requerido
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflar el diseño de este fragmento
        View rootViewgc = inflater.inflate( R.layout.fragment_gcorriente, container, false );
        String urlgc = "http://10.1.0.4:8080/Servidorade7880-1.0-SNAPSHOT/GraficoCorriente.jsp";
        WebView webvgc =  rootViewgc.findViewById( R.id.web_page_gc );
        webvgc.setWebViewClient( new WebViewClient() );
        webvgc.loadUrl( urlgc );
        return rootViewgc;
    }
}
```

MedicionesFragment.java

```
package com.tesis2020.equipodemediciontrifasico;

import android.os.Bundle;

import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.webkit.WebViewClient;

import java.util.Objects;


public class MedicionesFragment extends Fragment {

    public MedicionesFragment() {
        // Constructor público vacío requerido
    }
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflar el diseño de este fragmento
       View rootView = inflater.inflate( R.layout.fragment_mediciones, container, false );
       String url = "http://10.1.0.4:8080/Servidorade7880-1.0-SNAPSHOT/";
       WebView webv =  rootView.findViewById( R.id.web_page );
       webv.setWebViewClient( new WebViewClient() );
       webv.loadUrl( url );
       return rootView;
    }
}
```

GvoltajeFragment.java

```
package com.tesis2020.equipodemediciontrifasico;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.webkit.WebViewClient;


public class GvoltajeFragment extends Fragment {

    public GvoltajeFragment() {
        // Constructor público vacío requerido
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View rootViewgv = inflater.inflate( R.layout.fragment_gvoltaje, container, false );
        String urlgv = "http://10.1.0.4:8080/Servidorade7880-1.0-SNAPSHOT/GraficoVoltaje.jsp";
        WebView webvgv =  rootViewgv.findViewById( R.id.web_page_gv );
        webvgv.setWebViewClient( new WebViewClient() );
        webvgv.loadUrl( urlgv );
        return rootViewgv;
    }
}
```

GpactivaFragment.java

```
package com.tesis2020.equipodemediciontrifasico;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.webkit.WebViewClient;


public class GpactivaFragment extends Fragment {


    public GpactivaFragment() {
        // Constructor público vacío requerido
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View rootViewgpactiva = inflater.inflate( R.layout.fragment_gpactiva, container, false );
       String urlgpactiva = "http://10.1.0.4:8080/Servidorade7880-1.0-SNAPSHOT/GraficoPactiva.jsp";
        WebView webvgpactiva =  rootViewgpactiva.findViewById( R.id.web_page_gpactiva );
        webvgpactiva.setWebViewClient( new WebViewClient() );
        webvgpactiva.loadUrl( urlgpactiva );
        return rootViewgpactiva;
    }
}
```

AndroidManifes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tesis2020.equipodemediciontrifasico">
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.EQUIPODEMEDICIONTRIFASICO">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/Theme.EQUIPODEMEDICIONTRIFASICO.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Anexo F. Hoja de datos para los transformadores de corriente

Split core current transformer

YHDC®

Model：SCT013-100

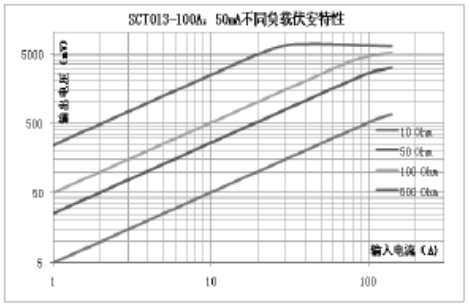
Characteristic:

Opening size 13mmx13mm, Compatible to foreign products, leading wire 1 metre, standardΦ3.5 three core plug output, current &voltage two types of output. (Patent No. ZL 2015 3 0060067.X）

Technical indicators:

Hanging installation,leading wire output
Fire resistance property:UL94-V0
Standard:GB1208-2006
Work temperature:-25℃～+70℃
Storage temperature:-30℃～+90℃
Work voltage:660V
Frequency range:50Hz-1KHz
Dielectric strength:3.5KV 50Hz 1min

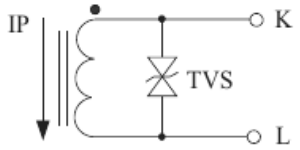
Characteristic curve in different load volt-ampere：



Electric parameter:

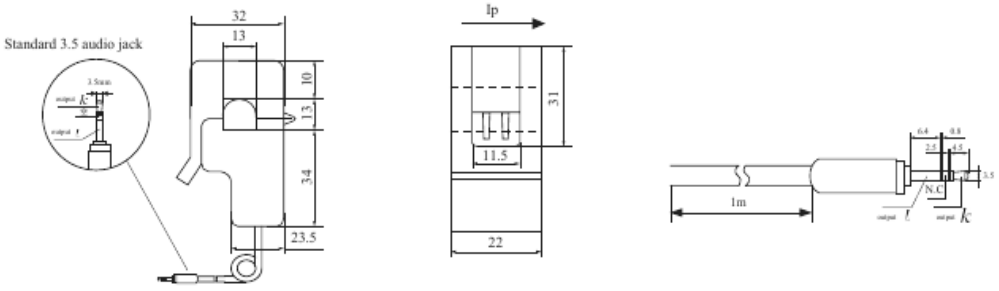
Rated input(rms)	100	A
Max.Input	120	A
Rated output	50	mA
Turns ratio	1:2000	
Accuracy	±1	%
Linearity	≤0.2	%
Phase error		
max:Sampling resistance	10	Ω
Weight	50	g

Wiring diagram:



Built-in two-way protection diode current output type

Outline size diagram(in mm):



Anexo G. Hoja de datos del circuito integrado ADE7880

ANALOG DEVICES

Polyphase Multifunction Energy Metering IC with Harmonic Monitoring

Data Sheet **ADE7880**

FEATURES

- **Highly accurate; supports IEC 62053-21, IEC 62053-22, IEC 62053-23, EN 50470-1, EN 50470-3, ANSI C12.20, and IEEE1459 standards**
- **Supports IEC 61000-4-7 Class I and Class II accuracy specification**
- **Compatible with 3-phase, 3-wire or 4-wire (delta or wye), and other 3-phase services**
- **Supplies rms, active, reactive, and apparent powers, power factor, THD, and harmonic distortion of all harmonics within 2.8 kHz pass band on all phases**
- **Supplies rms and harmonic distortions of all harmonics within 2.8 kHz pass band on neutral current**
- **Less than 1% error in harmonic current and voltage rms, harmonic active and reactive powers over a dynamic range of 2000 to 1 at T_A = 25°C**
- **Supplies total (fundamental and harmonic) active and apparent energy and fundamental active/reactive energy on each phase and on the overall system**
- **Less than 0.1% error in active and fundamental reactive energy over a dynamic range of 1000 to 1 at T_A = 25°C**
- **Less than 0.2% error in active and fundamental reactive energy over a dynamic range of 5000 to 1 at T_A = 25°C**
- **Less than 0.1% error in voltage and current rms over a dynamic range of 1000 to 1 at T_A = 25°C**
- **Battery supply input for missing neutral operation**
- **Wide supply voltage operation: 2.4 V to 3.7 V**
- **Reference: 1.2 V (drift 20 ppm/°C typical) with external overdrive capability**
- **40-lead lead frame chip scale package (LFCSP), Pb-free, pin-for-pin compatible with ADE7854, ADE7858, ADE7868 and ADE7878**

APPLICATIONS

- **Energy metering systems**
- **Power quality monitoring**
- **Solar inverters**
- **Process monitoring**
- **Protective devices**

GENERAL DESCRIPTION

The ADE7880[1] is a high accuracy, 3-phase electrical energy measurement IC with serial interfaces and three flexible pulse outputs. The ADE7880 device incorporates second-order sigma-delta (Σ-Δ) analog-to-digital converters (ADCs), a digital integrator, reference circuitry, and all of the signal processing required to perform the total (fundamental and harmonic) active, and apparent energy measurements, rms calculations, as well as fundamental-only active and reactive energy measurements. In addition, the ADE7880 computes the rms of harmonics on the phase and neutral currents and on the phase voltages, together with the active, reactive and apparent powers, and the power factor and harmonic distortion on each harmonic for all phases. Total harmonic distortion (THD) is computed for all currents and voltages. A fixed function digital signal processor (DSP) executes this signal processing. The DSP program is stored in the internal ROM memory.

The ADE7880 is suitable for measuring active, reactive, and apparent energy in various 3-phase configurations, such as wye or delta services with, both, three and four wires. The ADE7880 provides system calibration features for each phase, that is, rms offset correction, phase calibration, and gain calibration. The CF1, CF2, and CF3 logic outputs provide a wide choice of power information: total active powers, apparent powers, or the sum of the current rms values, and fundamental active and reactive powers.

The ADE7880 contains waveform sample registers that allow access to all ADC outputs. The devices also incorporate power quality measurements, such as short duration low or high voltage detections, short duration high current variations, line voltage period measurement, and angles between phase voltages and currents. Two serial interfaces, SPI and I²C, can be used to communicate with the ADE7880. A dedicated high speed interface, the high speed data capture (HSDC) port, can be used in conjunction with I²C to provide access to the ADC outputs and real-time power information. The ADE7880 also has two interrupt request pins, $\overline{\text{IRQ0}}$ and $\overline{\text{IRQ1}}$, to indicate that an enabled interrupt event has occurred. Three specially designed low power modes ensure the continuity of energy accumulation when the ADE7880 is in a tampering situation. The ADE7880 is available in the 40-lead LFCSP, Pb-free package, pin-for-pin compatible with ADE7854, ADE7858, ADE7868, and ADE7878 devices.

[1] Protected by U.S. Patent 8,010,304 B2. Other patents pending.

TABLE OF CONTENTS

REVISION HISTORY

12/14—Rev. B to Rev. C

Changes to Pin EP, Table 7 13
Changes to Configuring Harmonic Calculations Update Rate Section 66
Change to Address 0x43C7, Table 30 88
Changes to Bit 19, Table 36 94
Changes to Bit 19, Table 38 97

8/14—Rev. A to Rev. B

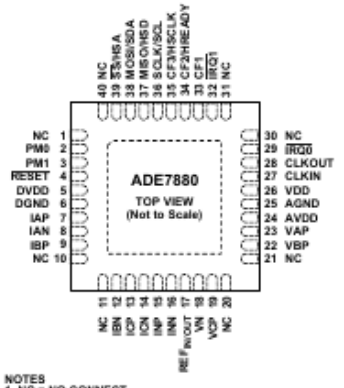
Change to Features Section 1
Changes to Patent Footnote 1
Changes to Functional Block Diagram 4
Changes to Table 1 5
Changes to Data Hold Time Parameter, Table 2, and Figure 2 8
Changes to Pin 5 and Pin 24, Table 7, and Figure 6 12
Changes to Figure 16 and Figure 18 15
Changes to Figure 20 and Figure 24 Caption 16
Moved Figure 29 and Figure 30 18
Changes to Test Circuit Section and Figure 32 19
Changes to Terminology Section 20
Changes to PSM2—Low Power Mode Section 21
Added Figure 34; Renumbered Sequentially 22
Change to Power-Up Procedure Section and Figure 35 24
Changes to Figure 42 and Figure 43 28
Changes to Figure 48 and Figure 50; Added Figure 49 30
Changes to Changing Phase Voltage Data Path Section and Figure 51 31
Changes to Power Quality Measurements Section and Figure 52 32
Changed ADC_{MAX} = 5,928,256, to ADC_{MAX} = 5,326,737, Neutral Current Mismatch Section 37
Added Figure 64 38
Changes to Reference Circuit Section and Digital Signal Processor Section 39
Changes to Current RMS Calculation Section 41
Changes to Voltage RMS Offset Compensation Section, Voltage RMS in 3-Phase, 3-Wire Delta Configurations Section, and Active Power Calculation Section 45
Changes to Figure 76 46
Changes to Fundamental Active Power Calculation Section 47
Added Managing Change in Fundamental Line Frequency Section 47
Changes to Figure 78 49
Changes to Active Energy Accumulation Modes Section 50
Changes to Fundamental Reactive Power Calculation Section and Equation 35 51
Changes to Fundamental Reactive Energy Accumulation Modes Section 54
Changes to Apparent Power Calculation Section 55
Changes to Apparent Energy Accumulation Modes Section and Figure 83 57
Changes to Power Factor Calculation Section and Harmonics Calculations Section 58
Changes to Figure 85 60
Changes to Energy-to-Frequency Conversion Section 66
Changes to Checksum Register Section, Equation 54, and Figure 100 73
Changes to Table 24 75
Changes to I^2C-Compatible Interface Section 76
Changes to Figure 109 80
Changes to ADE7880 Quick Setup as Energy Meter Section 82
Added Layout Guidelines Section 83
Added Crystal Circuit Section 84
Changes to Silicon Anomaly Section, Table 26, and Table 27 85
Changes to Table 30 87
Changes to Table 33 90
Changes to Bit 19, Table 36 94
Changes to Bit 19, Table 38 97
Changes to Table 42 99
Changes to Table 45 101
Changes to Table 50 104
Changes to Bits[4:3], Table 54 106

3/12—Rev. 0 to Rev. A

Removed References to + N (Plus Noise) and changed VTHDN to VTHD and ITHDN to ITHD ..... Throughout
Changes to Reactive Energy Management Parameter in Table 14
Changes to Figure 6 11
Changes to Table 7 12
Changes to Phase Compensation Section 36
Changes to Equation 13 39
Changes to Equation 33 49
Changes to Fundamental Reactive Energy Calculation Section 51
Changes to Figure 80 55
Changes to Figure 85 62
Changes to Energy Registers and CF Outputs for Various Accumulation Modes Section 67
Changes to Figure 95 69
Changes to No Load Condition Section 69
Changes to Equation 53 71
Changes to Figure 100 74
Changes to Figure 101 and to Figure 102 75
Changes to SPI-Compatible Interface Section 76
Changes to HSDC Interface Section 78
Changes to Figure 109 and to Figure 110 80
Changes to Silicon Anomaly Section 81
Changes to Table 48 99
Changes to Table 52 101

10/11—Revision 0: Initial Version

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS



NOTES
1. NC = NO CONNECT.
2. CREATE A SIMILAR PAD ON THE PCB UNDER THE EXPOSED PAD. SOLDER THE EXPOSED PAD TO THE PAD ON THE PCB TO CONFER MECHANICAL STRENGTH TO THE PACKAGE. CONNECT THE PADS TO AGND AND DGND.

Figure 6. Pin Configuration

Table 7. Pin Function Descriptions

Pin No.	Mnemonic	Description
1, 10, 11, 20, 21, 30, 31, 40	NC	No Connect. Do not connect to these pins. These pins are not connected internally.
2	PM0	Power Mode Pin 0. This pin, combined with PM1, defines the power mode of the ADE7880, as described in Table 8.
3	PM1	Power Mode Pin 1. This pin defines the power mode of the ADE7880 when combined with PM0, as described in Table 8.
4	$\overline{\text{RESET}}$	Reset Input, Active Low. In PSM0 mode, this pin must stay low for at least 10 µs to trigger a hardware reset.
5	DVDD	2.5 V Output of the Digital Low Dropout (LDO) Regulator. Decouple this pin with a 4.7 µF capacitor in parallel with a ceramic 220 nF capacitor. Do not connect external active circuitry to this pin.
6	DGND	Ground Reference. This pin provides the ground reference for the digital circuitry.
7, 8	IAP, IAN	Analog Inputs for Current Channel A. This channel is used with the current transducers and is referenced in this data sheet as Current Channel A. These inputs are fully differential voltage inputs with a maximum differential level of ±0.5 V. This channel also has an internal PGA equal to the ones on Channel B and Channel C.
9, 12	IBP, IBN	Analog Inputs for Current Channel B. This channel is used with the current transducers and is referenced in this data sheet as Current Channel B. These inputs are fully differential voltage inputs with a maximum differential level of ±0.5 V. This channel also has an internal PGA equal to the ones on Channel C and Channel A.
13, 14	ICP, ICN	Analog Inputs for Current Channel C. This channel is used with the current transducers and is referenced in this data sheet as Current Channel C. These inputs are fully differential voltage inputs with a maximum differential level of ±0.5 V. This channel also has an internal PGA equal to the ones on Channel A and Channel B.
15, 16	INP, INN	Analog Inputs for Neutral Current Channel N. This channel is used with the current transducers and is referenced in this data sheet as Current Channel N. These inputs are fully differential voltage inputs with a maximum differential level of ±0.5 V. This channel also has an internal PGA, different from the ones found on the A, B, and C channels.
17	$\text{REF}_{\text{IN/OUT}}$	This pin provides access to the on-chip voltage reference. The on-chip reference has a nominal value of 1.2 V. An external reference source with 1.2 V ± 8% can also be connected at this pin. In either case, decouple this pin to AGND with a 4.7 µF capacitor in parallel with a ceramic 100 nF capacitor. After reset, the on-chip reference is enabled.

Pin No.	Mnemonic	Description
18, 19, 22, 23	VN, VCP, VBP, VAP	Analog Inputs for the Voltage Channel. This channel is used with the voltage transducer and is referenced as the voltage channel in this data sheet. These inputs are single-ended voltage inputs with a maximum signal level of ±0.5 V with respect to VN for specified operation. This channel also has an internal PGA.
24	AVDD	2.5 V Output of the Analog Low Dropout (LDO) Regulator. Decouple this pin with a 4.7 µF capacitor in parallel with a ceramic 220 nF capacitor. Do not connect external active circuitry to this pin.
25	AGND	Ground Reference. This pin provides the ground reference for the analog circuitry. Tie this pin to the analog ground plane or to the quietest ground reference in the system. Use this quiet ground reference for all analog circuitry, for example, antialiasing filters, current, and voltage transducers.
26	VDD	Supply Voltage. This pin provides the supply voltage. In PSM0 (normal power mode), maintain the supply voltage at 3.3 V ± 10% for specified operation. In PSM1 (reduced power mode), PSM2 (low power mode), and PSM3 (sleep mode), when the ADE7880 is supplied from a battery, maintain the supply voltage between 2.4 V and 3.7 V. Decouple this pin to DGND with a 10 µF capacitor in parallel with a ceramic 100 nF capacitor.
27	CLKIN	Master Clock. An external clock can be provided at this logic input. Alternatively, a parallel resonant AT-cut crystal can be connected across CLKIN and CLKOUT to provide a clock source for the ADE7880. The clock frequency for specified operation is 16.384 MHz. Use ceramic load capacitors of a few tens of picofarad with the gate oscillator circuit. Refer to the data sheet of the crystal manufacturer for load capacitance requirements.
28	CLKOUT	A crystal can be connected across this pin and CLKIN (as previously described with Pin 27 in this table) to provide a clock source for the ADE7880.
29, 32	$\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$	Interrupt Request Outputs. These are active low logic outputs. See the Interrupts section for a detailed presentation of the events that can trigger interrupts.
33, 34, 35	CF1, CF2/HREADY, CF3/HSCLK	Calibration Frequency (CF) Logic Outputs. These outputs provide power information based on the CF1SEL[2:0], CF2SEL[2:0], and CF3SEL[2:0] bits in the CFMODE register. These outputs are used for operational and calibration purposes. The full-scale output frequency can be scaled by writing to the CF1DEN, CF2DEN, and CF3DEN registers, respectively (see the Energy-to-Frequency Conversion section). CF2 is multiplexed with the HREADY signal generated by the harmonic calculations block. CF3 is multiplexed with the serial clock output of the HSDC port.
36	SCLK/SCL	Serial Clock Input for SPI Port/Serial Clock Input for I²C Port. All serial data transfers are synchronized to this clock (see the Serial Interfaces section). This pin has a Schmidt trigger input for use with a clock source that has a slow edge transition time, for example, optoisolator outputs.
37	MISO/HSD	Data Out for SPI Port/Data Out for HSDC Port.
38	MOSI/SDA	Data In for SPI Port/Data Out for I²C Port.
39	$\overline{\text{SS}}$/HSA	Slave Select for SPI Port/HSDC Port Active.
EP	Exposed Pad	Create a similar pad on the PCB under the exposed pad. Solder the exposed pad to the pad on the PCB to confer mechanical strength to the package. Connect the pads to AGND and DGND.

TEST CIRCUIT

In Figure 32, the PM1 and PM0 pins are pulled up internally to VDD. Select the mode of operation by using a microcontroller to programmatically change the pin values. See the Power Management section for details.

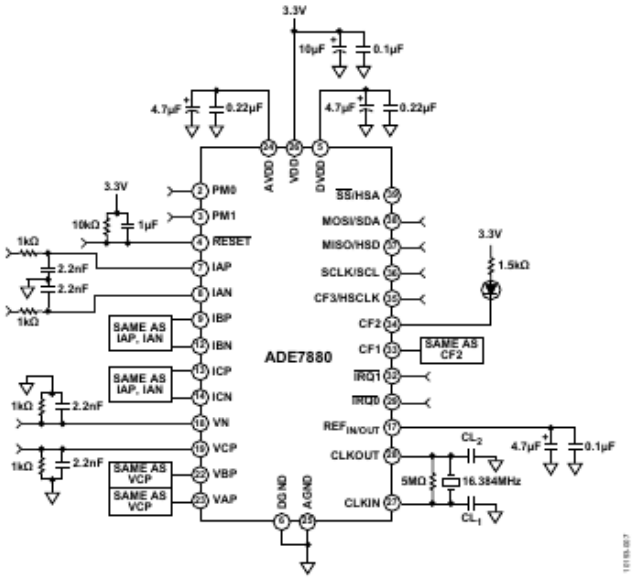


Figure 32. Test Circuit

LAYOUT GUIDELINES

Figure 114 presents a basic schematic of the ADE7880 together with its surrounding circuitry: decoupling capacitors at pins VDD, AVDD, DVDD, and $REF_{IN/OUT}$, the 16.384 MHz crystal, and its load capacitors. The rest of the pins are dependent on the particular application and are not shown here.

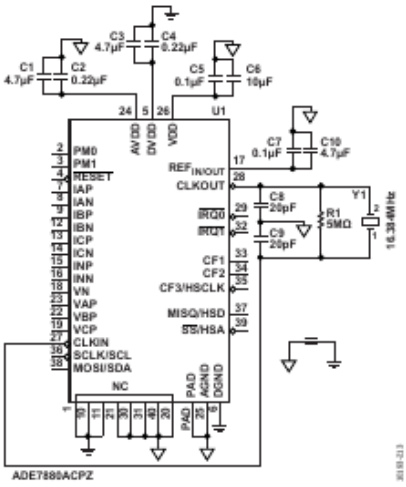


Figure 114. ADE7880 Crystal and Capacitors Selection

Figure 115 and Figure 116 present a proposed layout of a printed circuit board (PCB) with two layers that have the components placed only on the top of the board. Following these layout guidelines helps in creating a low noise design with higher immunity to EMC influences.

The VDD, AVDD, DVDD, and $REF_{IN/OUT}$ pins have two decoupling capacitors each, one of µF order and a ceramic one of 220 nF or 100 nF. These ceramic capacitors need to be placed the closest to the ADE7880 as they decouple high frequency noises, while the µF ones must be placed in close proximity.

The crystal load capacitors need to be placed closest to the ADE7880, while the crystal can be placed in close proximity.

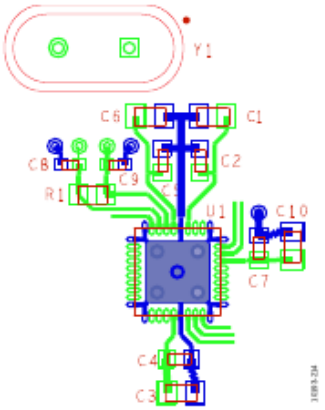


Figure 115. ADE7880 Top Layer Printed Circuit Board

The exposed pad of the ADE7880 is soldered to an equivalent pad on the PCB. The AGND and DGND traces of the ADE7880 are then routed directly into the PCB pad.

The bottom layer is composed mainly of a ground plane surrounding as much as possible the crystal traces.

REGISTERS LIST

Table 30. Registers Located in DSP Data Memory RAM

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value	Description
0x4380	AIGAIN	R/W	24	32 ZPSE	S	0x000000	Phase A current gain adjust.
0x4381	AVGAIN	R/W	24	32 ZPSE	S	0x000000	Phase A voltage gain adjust.
0x4382	BIGAIN	R/W	24	32 ZPSE	S	0x000000	Phase B current gain adjust.
0x4383	BVGAIN	R/W	24	32 ZPSE	S	0x000000	Phase B voltage gain adjust.
0x4384	CIGAIN	R/W	24	32 ZPSE	S	0x000000	Phase C current gain adjust.
0x4385	CVGAIN	R/W	24	32 ZPSE	S	0x000000	Phase C voltage gain adjust.
0x4386	NIGAIN	R/W	24	32 ZPSE	S	0x000000	Neutral current gain adjust.
0x4387	Reserved	R/W	24	32 ZPSE	S	0x000000	Do not write this location for proper operation.
0x4388	DICOEFF	R/W	24	32 ZPSE	S	0x0000000	Register used in the digital integrator algorithm. If the integrator is turned on, it must be set at 0xFF8000. In practice, it is transmitted as 0xFFF8000.
0x4389	APGAIN	R/W	24	32 ZPSE	S	0x000000	Phase A power gain adjust.
0x438A	AWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase A total active power offset adjust.
0x438B	BPGAIN	R/W	24	32 ZPSE	S	0x000000	Phase B power gain adjust.
0x438C	BWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase B total active power offset adjust.
0x438D	CPGAIN	R/W	24	32 ZPSE	S	0x000000	Phase C power gain adjust.
0x438E	CWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase C total active power offset adjust.
0x438F	AIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase A current rms offset.
0x4390	AVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase A voltage rms offset.
0x4391	BIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase B current rms offset.
0x4392	BVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase B voltage rms offset.
0x4393	CIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase C current rms offset.
0x4394	CVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase C voltage rms offset.
0x4395	NIRMSOS	R/W	24	32 ZPSE	S	0x000000	Neutral current rms offset.
0x4396-0x4397	Reserved	N/A	N/A	N/A	N/A	0x000000	Do not write these memory locations for proper operation.
0x4398	HPGAIN	R/W	24	32 ZPSE	S	0x000000	Harmonic powers gain adjust.
0x4399	ISUMLVL	R/W	24	32 ZPSE	S	0x000000	Threshold used in comparison between the sum of phase currents and the neutral current.
0x439A-0x439E	Reserved	N/A	N/A	N/A	N/A	0x000000	Do not write these memory locations for proper operation.
0x439F	VLEVEL	R/W	28	32 ZP	S	0x0000000	Register used in the algorithm that computes the fundamental active and reactive powers. Set this register according to Equation 22 for proper functioning of fundamental powers and harmonic computations.
0x43A0-0x43A1	Reserved	N/A	N/A	N/A	N/A	0x000000	Do not write these memory locations for proper operation.
0x43A2	AFWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase A fundamental active power offset adjust.
0x43A3	BFWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase B fundamental active power offset adjust.
0x43A4	CFWATTOS	R/W	24	32 ZPSE	S	0x000000	Phase C fundamental active power offset adjust.
0x43A5	AFVAROS	R/W	24	32 ZPSE	S	0x000000	Phase A fundamental reactive power offset adjust.
0x43A6	BFVAROS	R/W	24	32 ZPSE	S	0x000000	Phase B fundamental reactive power offset adjust.
0x43A7	CFVAROS	R/W	24	32 ZPSE	S	0x000000	Phase C fundamental reactive power offset adjust.
0x43A8	AFIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase A fundamental current rms offset.
0x43A9	BFIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase B fundamental current rms offset.
0x43AA	CFIRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase C fundamental current rms offset.
0x43AB	AFVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase A fundamental voltage rms offset.
0x43AC	BFVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase B fundamental voltage rms offset.
0x43AD	CFVRMSOS	R/W	24	32 ZPSE	S	0x000000	Phase C fundamental voltage rms offset.
0x43AE	HXWATTOS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic X (see Harmonics Calculations section for details).

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value	Description
0x43AF	HYWATTOS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic Y (see Harmonics Calculations section for details).
0x43B0	HZWATTOS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic Z (see Harmonics Calculations section for details).
0x43B1	HXVAROS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic X (see Harmonics Calculations section for details).
0x43B2	HYVAROS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic Y (see Harmonics Calculations section for details).
0x43B3	HZVAROS	R/W	24	32 ZPSE	S	0x000000	Active power offset adjust on harmonic Z (see Harmonics Calculations section for details).
0x43B4	HXIRMSOS	R/W	24	32 ZPSE	S	0x000000	Current rms offset on harmonic X (see Harmonics Calculations section for details).
0x43B5	HYIRMSOS	R/W	24	32 ZPSE	S	0x000000	Current rms offset on harmonic Y (see Harmonics Calculations section for details).
0x43B6	HZIRMSOS	R/W	24	32 ZPSE	S	0x000000	Current rms offset on harmonic Z (see Harmonics Calculations section for details).
0x43B7	HXVRMSOS	R/W	24	32 ZPSE	S	0x000000	Voltage rms offset on harmonic X (see Harmonics Calculations section for details).
0x43B8	HYVRMSOS	R/W	24	32 ZPSE	S	0x000000	Voltage rms offset on harmonic Y (see Harmonics Calculations section for details).
0x43B9	HZVRMSOS	R/W	24	32 ZPSE	S	0x000000	Voltage rms offset on harmonic Z (see Harmonics Calculations section for details).
0x43BA to 0x43BF	Reserved	N/A	N/A	N/A	N/A	0x000000	Do not write these memory locations for proper operation.
0x43C0	AIRMS	R	24	32 ZP	S	N/A	Phase A current rms value.
0x43C1	AVRMS	R	24	32 ZP	S	N/A	Phase A voltage rms value.
0x43C2	BIRMS	R	24	32 ZP	S	N/A	Phase B current rms value.
0x43C3	BVRMS	R	24	32 ZP	S	N/A	Phase B voltage rms value.
0x43C4	CIRMS	R	24	32 ZP	S	N/A	Phase C current rms value.
0x43C5	CVRMS	R	24	32 ZP	S	N/A	Phase C voltage rms value.
0x43C6	NIRMS	R	24	32 ZP	S	N/A	Neutral current rms value.
0x43C7	ISUM	R	28	32 ZP	S	N/A	Sum of IAWV, IBWV and ICWV registers.
0x43C8 to 0x43FF	Reserved	N/A	N/A	N/A	N/A	N/A	Do not write these memory locations for proper operation.

[1] R is read, and W is write.
[2] 32 ZPSE = 24-bit signed register that is transmitted as a 32-bit word with four MSBs padded with 0s and sign extended to 28 bits. Whereas 32 ZP = 28- bit or 24-bit signed or unsigned register that is transmitted as a 32-bit word with four or eight MSBs, respectively, padded with 0s.
[3] U is unsigned register, and S is signed register in twos complement format.

Table 31. Internal DSP Memory RAM Registers

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication	Type[2]	Default Value	Description
0xE203	Reserved	R/W	16	16	U	0x0000	Do not write this memory location for proper operation.
0xE228	Run	R/W	16	16	U	0x0000	Run register starts and stops the DSP. See the Digital Signal Processor section for more details.

[1] R is read, and W is write.
[2] U is unsigned register, and S is signed register in twos complement format.

Table 32. Billable Registers

Address	Register Name	R/W[1, 2]	Bit Length[2]	Bit Length During Communication[2]	Type[2, 3]	Default Value	Description
0xE400	AWATTHR	R	32	32	S	0x00000000	Phase A total active energy accumulation.
0xE401	BWATTHR	R	32	32	S	0x00000000	Phase B total active energy accumulation.
0xE402	CWATTHR	R	32	32	S	0x00000000	Phase C total active energy accumulation.
0xE403	AFWATTHR	R	32	32	S	0x00000000	Phase A fundamental active energy accumulation.
0xE404	BFWATTHR	R	32	32	S	0x00000000	Phase B fundamental active energy accumulation.
0xE405	CFWATTHR	R	32	32	S	0x00000000	Phase C fundamental active energy accumulation.
0xE406 to 0xE408	Reserved	R	32	32	S	0x00000000	
0xE409	AFVARHR	R	32	32	S	0x00000000	Phase A fundamental reactive energy accumulation.
0xE40A	BFVARHR	R	32	32	S	0x00000000	Phase B fundamental reactive energy accumulation.
0xE40B	CFVARHR	R	32	32	S	0x00000000	Phase C fundamental reactive energy accumulation.
0xE40C	AVAHR	R	32	32	S	0x00000000	Phase A apparent energy accumulation.
0xE40D	BVAHR	R	32	32	S	0x00000000	Phase B apparent energy accumulation.
0xE40E	CVAHR	R	32	32	S	0x00000000	Phase C apparent energy accumulation.

[1] R is read, and W is write.
[2] N/A is not applicable.
[3] U is unsigned register, and S is signed register in twos complement format.

Table 33. Configuration and Power Quality Registers

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value[4]	Description
0xE500	IPEAK	R	32	32	U	N/A	Current peak register. See Figure 60 and Table 34 for details about its composition.
0xE501	VPEAK	R	32	32	U	N/A	Voltage peak register. See Figure 60 and Table 35 for details about its composition.
0xE502	STATUS0	R/W	32	32	U	N/A	Interrupt Status Register 0. See Table 36.
0xE503	STATUS1	R/W	32	32	U	N/A	Interrupt Status Register 1. See Table 37.
0xE504	AIMAV	R	20	32 ZP	U	N/A	Phase A current mean absolute value computed during PSM0 and PSM1 modes.
0xE505	BIMAV	R	20	32 ZP	U	N/A	Phase B current mean absolute value computed during PSM0 and PSM1 modes.
0xE506	CIMAV	R	20	32 ZP	U	N/A	Phase C current mean absolute value computed during PSM0 and PSM1 modes.
0xE507	OILVL	R/W	24	32 ZP	U	0xFFFFFF	Overcurrent threshold.
0xE508	OVLVL	R/W	24	32 ZP	U	0xFFFFFF	Overvoltage threshold.
0xE509	SAGLVL	R/W	24	32 ZP	U	0x000000	Voltage SAG level threshold.
0xE50A	MASK0	R/W	32	32	U	0x00000000	Interrupt Enable Register 0. See Table 38.
0xE50B	MASK1	R/W	32	32	U	0x00000000	Interrupt Enable Register 1. See Table 39.
0xE50C	IAWV	R	24	32 SE	S	N/A	Instantaneous value of Phase A current.
0xE50D	IBWV	R	24	32 SE	S	N/A	Instantaneous value of Phase B current.
0xE50E	ICWV	R	24	32 SE	S	N/A	Instantaneous value of Phase C current.
0xE50F	INWV	R	24	32 SE	S	N/A	Instantaneous value of neutral current.
0xE510	VAWV	R	24	32 SE	S	N/A	Instantaneous value of Phase A voltage.
0xE511	VBWV	R	24	32 SE	S	N/A	Instantaneous value of Phase B voltage.
0xE512	VCWV	R	24	32 SE	S	N/A	Instantaneous value of Phase C voltage.

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value[4]	Description
0xE513	AWATT	R	24	32 SE	S	N/A	Instantaneous value of Phase A total active power.
0xE514	BWATT	R	24	32 SE	S	N/A	Instantaneous value of Phase B total active power.
0xE515	CWATT	R	24	32 SE	S	N/A	Instantaneous value of Phase C total active power.
0xE516 to 0xE518	Reserved	R	24	32 SE	S	0x000000	
0xE519	AVA	R	24	32 SE	S	N/A	Instantaneous value of Phase A apparent power.
0xE51A	BVA	R	24	32 SE	S	N/A	Instantaneous value of Phase B apparent power.
0xE51B	CVA	R	24	32 SE	S	N/A	Instantaneous value of Phase C apparent power.
0xE51F	CHECKSUM	R	32	32	U	0xAFFA63B9	Checksum verification. See the Checksum Register section for details.
0xE520	VNOM	R/W	24	32 ZP	S	0x000000	Nominal phase voltage rms used in the alternative computation of the apparent power. When the VNOMxEN bit is set, the applied voltage input in the corresponding phase is ignored and all corresponding rms voltage instances are replaced by the value in the VNOM register.
0xE521 to 0xE5FE	Reserved						Do not write these addresses for proper operation.
0xE5FF	LAST_RWDATA32	R	32	32	U	N/A	Contains the data from the last successful 32-bit register communication.
0xE600	PHSTATUS	R	16	16	U	N/A	Phase peak register. See Table 40.
0xE601	ANGLE0	R	16	16	U	N/A	Time Delay 0. See the Time Interval Between Phases section for details.
0xE602	ANGLE1	R	16	16	U	N/A	Time Delay 1. See the Time Interval Between Phases section for details.
0xE603	ANGLE2	R	16	16	U	N/A	Time Delay 2. See the Time Interval Between Phases section for details.
0xE604 to 0xE607	Reserved						Do not write these addresses for proper operation.
0xE608	PHNOLOAD	R	16	16	U	N/A	Phase no load register. See Table 41.
0xE609 to 0xE60B	Reserved						Do not write these addresses for proper operation.
0xE60C	LINECYC	R/W	16	16	U	0xFFFF	Line cycle accumulation mode count.
0xE60D	ZXTOUT	R/W	16	16	U	0xFFFF	Zero-crossing timeout count.
0xE60E	COMPMODE	R/W	16	16	U	0x01FF	Computation-mode register. See Table 42.
0xE60F	Gain	R/W	16	16	U	0x0000	PGA gains at ADC inputs. See Table 43.
0xE610	CFMODE	R/W	16	16	U	0x0EA0	CFx configuration register. See Table 44.
0xE611	CF1DEN	R/W	16	16	U	0x0000	CF1 denominator.
0xE612	CF2DEN	R/W	16	16	U	0x0000	CF2 denominator.
0xE613	CF3DEN	R/W	16	16	U	0x0000	CF3 denominator.
0xE614	APHCAL	R/W	10	16 ZP	S	0x0000	Phase calibration of Phase A. See Table 45.
0xE615	BPHCAL	R/W	10	16 ZP	S	0x0000	Phase calibration of Phase B. See Table 45.
0xE616	CPHCAL	R/W	10	16 ZP	S	0x0000	Phase calibration Phase of C. See Table 45.
0xE617	PHSIGN	R	16	16	U	N/A	Power sign register. See Table 46.
0xE618	CONFIG	R/W	16	16	U	0x0002	ADE7880 configuration register. See Table 47.
0xE700	MMODE	R/W	8	8	U	0x1C	Measurement mode register. See Table 48.

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value[4]	Description
0xE701	ACCMODE	R/W	8	8	U	0x80	Accumulation mode register. See Table 49.
0xE702	LCYCMODE	R/W	8	8	U	0x78	Line accumulation mode behavior. See Table 51.
0xE703	PEAKCYC	R/W	8	8	U	0x00	Peak detection half line cycles.
0xE704	SAGCYC	R/W	8	8	U	0x00	SAG detection half line cycles.
0xE705	CFCYC	R/W	8	8	U	0x01	Number of CF pulses between two consecutive energy latches. See the Synchronizing Energy Registers with CFx Outputs section.
0xE706	HSDC_CFG	R/W	8	8	U	0x00	HSDC configuration register. See Table 52.
0xE707	Version	R	8	8	U		Version of die.
0xE7E4	Reserved	R	8	8	U	0x08	This register must remain at this value for checksum functionality to work. If this register shows a different value while being read, reset the chip before working with the checksum feature.
0xE7FD	LAST_RWDATA8	R	8	8	U	N/A	Contains the data from the last successful 8-bit register communication.
0xE880	FVRMS	R	24	32	S	N/A	The rms value of the fundamental component of the phase voltage.
0xE881	FIRMS	R	24	32	S	N/A	The rms value of the fundamental component of the phase current
0xE882	FWATT	R	24	32	S	N/A	The active power of the fundamental component.
0xE883	FVAR	R	24	32	S	N/A	The reactive power of the fundamental component.
0xE884	FVA	R	24	32	S	N/A	The apparent power of the fundamental component.
0xE885	FPF	R	24	32	S	N/A	The power factor of the fundamental component.
0xE886	VTHD	R	24	32	S	N/A	Total harmonic distortion of the phase voltage.
0xE887	ITHD	R	24	32	S	N/A	Total harmonic distortion of the phase current.
0xE888	HXVRMS	R	24	32	S	N/A	The rms value of the phase voltage harmonic X.
0xE889	HXIRMS	R	24	32	S	N/A	The rms value of the phase current harmonic X.
0xE88A	HXWATT	R	24	32	S	N/A	The active power of the harmonic X.
0xE88B	HXVAR	R	24	32	S	N/A	The reactive power of the harmonic X.
0xE88C	HXVA	R	24	32	S	N/A	The apparent power of the harmonic X.
0xE88D	HXPF	R	24	32	S	N/A	The power factor of the harmonic X.
0xE88E	HXVHD	R	24	32	S	N/A	Harmonic distortion of the phase voltage harmonic X relative to the fundamental.
0xE88F	HXIHD	R	24	32	S	N/A	Harmonic distortion of the phase current harmonic X relative to the fundamental.
0xE890	HYVRMS	R	24	32	S	N/A	The rms value of the phase voltage harmonic Y.
0xE891	HYIRMS	R	24	32	S	N/A	The rms value of the phase current harmonic Y.
0xE892	HYWATT	R	24	32	S	N/A	The active power of the harmonic Y.
0xE893	HYVAR	R	24	32	S	N/A	The reactive power of the harmonic Y.
0xE894	HYVA	R	24	32	S	N/A	The apparent power of the harmonic Y.
0xE895	HYPF	R	24	32	S	N/A	The power factor of the harmonic Y.

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value[4]	Description
0xE896	HYVHD	R	24	32	S	N/A	Harmonic distortion of the phase voltage harmonic Y relative to the fundamental.
0xE897	HYIHD	R	24	32	S	N/A	Harmonic distortion of the phase current harmonic Y relative to the fundamental.
0xE898	HZVRMS	R	24	32	S	N/A	The rms value of the phase voltage harmonic Z.
0xE899	HZIRMS	R	24	32	S	N/A	The rms value of the phase current harmonic Z.
0xE89A	HZWATT	R	24	32	S	N/A	The active power of the harmonic Z.
0xE89B	HZVAR	R	24	32	S	N/A	The reactive power of the harmonic Z.
0xE89C	HZVA	R	24	32	S	N/A	The apparent power of the harmonic Z.
0xE89D	HZPF	R	24	32	S	N/A	The power factor of the harmonic Z.
0xE89E	HZVHD	R	24	32	S	N/A	Harmonic distortion of the phase voltage harmonic Z relative to the fundamental.
0xE89F	HZIHD	R	24	32	S	N/A	Harmonic distortion of the phase current harmonic Z relative to the fundamental.
0xE8A0 to 0xE8FF	Reserved		24	32			Reserved. These registers are always 0.
0xE900	HCONFIG	R/W	16	16	U	0x08	Harmonic Calculations Configuration register. See Table 54.
0xE902	APF	R	16	16	S	N/A	Phase A power factor.
0xE903	BPF	R	16	16	S	N/A	Phase B power factor.
0xE904	CPF	R	16	16	S	N/A	Phase C power factor.
0xE905	APERIOD	R	16	16	U	N/A	Line period on Phase A voltage.
0xE906	BPERIOD	R	16	16	U	N/A	Line period on Phase B voltage.
0xE907	CPERIOD	R	16	16	U	N/A	Line period on Phase C voltage.
0xE908	APNOLOAD	R/W	16	16	U	0x0000	No load threshold in the total/ fundamental active power data paths. Do not write 0xFFFF to this register.
0xE909	VARNOLOAD	R/W	16	16	U	0x0000	No load threshold in the total/ fundamental reactive power data path. Do not write 0xFFFF to this register.
0xE90A	VANOLOAD	R/W	16	16	U	0x0000	No load threshold in the apparent power data path. Do not write 0xFFFF to this register.
0xE9FE	LAST_ADD	R	16	16	U	N/A	The address of the register successfully accessed during the last read/write operation.
0xE9FF	LAST_RWDATA16	R	16	16	U	N/A	Contains the data from the last successful 16-bit register communication.
0xEA00	CONFIG3	R/W	8	8	U	0x01	Configuration register. See Table 53.
0xEA01	LAST_OP	R	8	8	U	N/A	Indicates the type, read or write, of the last successful read/write operation.
0xEA02	WTHR	R/W	8	8	U	0x03	Threshold used in phase total/ fundamental active power data path.
0xEA03	VARTHR	R/W	8	8	U	0x03	Threshold used in phase total/ fundamental reactive power data path.
0xEA04	VATHR	R/W	8	8	U	0x03	Threshold used in phase apparent power data path.
0xEA05 to 0xEA07	Reserved		8	8			Reserved. These registers are always 0.
0xEA08	HX	R/W	8	8	U	3	Selects an index of the harmonic monitored by the harmonic computations.
0xEA09	HY	R/W	8	8	U	5	Selects an index of the harmonic monitored by the harmonic computations.

Address	Register Name	R/W[1]	Bit Length	Bit Length During Communication[2]	Type[3]	Default Value[4]	Description
0xEA0A	HZ	R/W	8	8	U	7	Selects an index of the harmonic monitored by the harmonic computations.
0xEA0B to 0xEBFE	Reserved		8	8			Reserved. These registers are always 0.
0xEBFF	Reserved		8	8			This address can be used in manipulating the $\overline{SS}$/HSA pin when SPI is chosen as the active port. See the Serial Interfaces section for details.
0xEC00	LPOILVL	R/W	8	8	U	0x07	Overcurrent threshold used during PSM2 mode. See Table 55 in which the register is detailed.
0xEC01	CONFIG2	R/W	8	8	U	0x00	Configuration register used during PSM1 mode. See Table 56.

[1] R is read, and W is write.
[2] 32 ZP = 24- or 20-bit signed or unsigned register that is transmitted as a 32-bit word with 8 or 12 MSBs, respectively, padded with 0s. 32 SE = 24-bit signed register that is transmitted as a 32-bit word sign extended to 32 bits. 16 ZP = 10-bit unsigned register that is transmitted as a 16-bit word with six MSBs padded with 0s.
[3] U is unsigned register, and S is signed register in twos complement format.
[4] N/A is not applicable.

OUTLINE DIMENSIONS

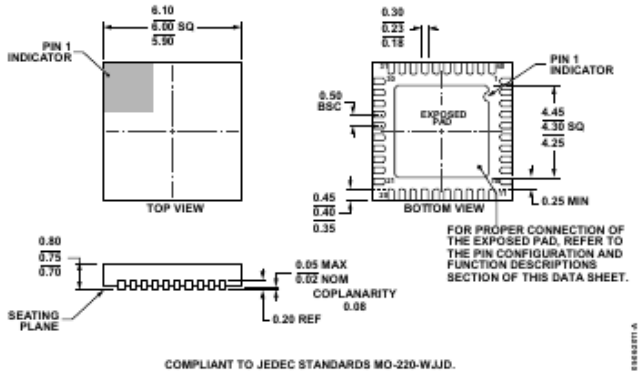


Figure 118. 40-Lead Lead Frame Chip Scale Package [LFCSP_WQ]
6 mm × 6 mm Body, Very Very Thin Quad
(CP-40-10)
Dimensions shown in millimeters

ORDERING GUIDE

Model[1]	Temperature Range	Package Description	Package Option
ADE7880ACPZ	−40°C to +85°C	40-Lead LFCSP_WQ	CP-40-10
ADE7880ACPZ-RL	−40°C to +85°C	40-Lead LFCSP_WQ, 13" Tape and Reel	CP-40-10
EVAL-ADE7880EBZ		Evaluation Board	

[1] Z = RoHS Compliant Part.

Anexo H. Calibración y pruebas de exactitud del medidor ADE7880

CALIBRACIÓN DEL MEDIDOR DEDICADO ADE7880

Para obtener lecturas precisa provenientes del CI ADE7880 este requiere ser calibrado. La calibración es necesaria para cada medidor basado en el CI ADE7880.

Al diseñar un medidor utilizando el CI ADE7880, se requiere de un máximo de tres etapas de calibración las cuales incluyen a la ganancia, la fase y el desplazamiento. Dependiendo de la configuración externa y la clase del medidor se pudiera omitir alguna de estas etapas de calibración. La tabla 10 proporciona información sobre qué pasos de calibración se requieren normalmente para una configuración en particular. Dado que los requisitos y el rendimiento puede diferir de un diseño a otro. El rendimiento del medidor debe evaluarse para determinar si se requieren pasos de calibración adicionales.

Etapa de calibración	Requisito típico
Ganancia.	Siempre es un requisito.
Fase.	Cuando se utiliza un sensor de corriente que introduce un retardo de fase, como un transformador de corriente (CT) o una bobina Rogowski, a menudo se requiere. Cuando se utiliza un sensor de corriente que no introduce un retraso, normalmente no es necesario.
Desplazamiento.	Cuando se busca alta precisión en un amplio rango dinámico, a menudo se requiere. Por lo general no se requiere para todos los diseños de medidores.

Tabla 10: Etapas para la calibración.

Si en el diseño del medidor se requiere calibración a una constante del medidor en particular, se suele utilizar el pin de salida CFx. De no utilizar la salida CFx y no utilizar una constante del medidor por diseño, el registro puede ser un método más conveniente. Los registros de energía resultan en lecturas precisas en el pin de salida CFx y viceversa. Ambos métodos resultan en el mismo nivel de precisión.

METODOS DE CALIBRACION

Se pueden utilizar dos tipos de equipos para calibrar el CI ADE7880:

- Un medidor de referencia.
- Una fuente precisa.

MEDIDOR DE REFERENCIA

Si se utiliza el método por medidor de referencia debe utilizarse en consecuencia las salidas CFx debido a que el medidor de referencia determina el error basado en la frecuencia con la que se emiten los pulsos. El medidor de referencia debe ser más preciso que el medidor

resultante. La conexión de un medidor de referencia y un equipo de medición a ser calibrado se muestra en la figura 111.

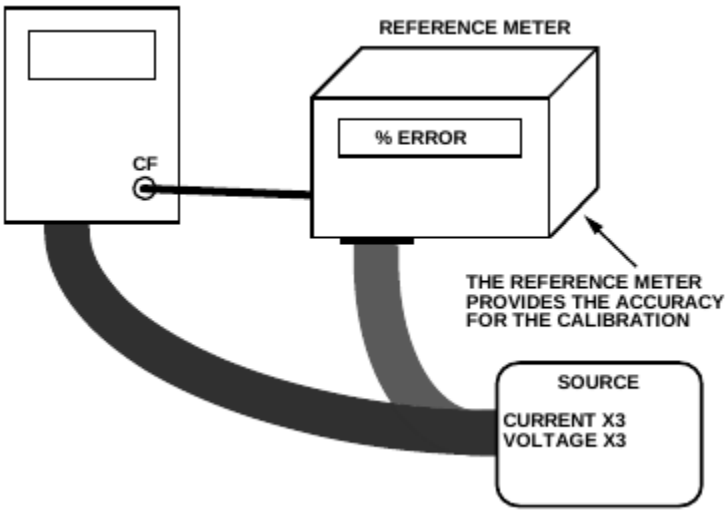


Figura 111: Conexiones del medidor de referencia.

FUENTE PRECISA

El segundo método de calibración es utilizar una fuente precisa para realizar la calibración. Si se utiliza este método, se puede utilizar la salida CFx o los registros de energía para acceder a los datos de energía. La fuente precisa debe ser capaz de proporcionar una entrada de tensión y corriente controlable con mayor precisión que la requerida en el medidor resultante. La figura 112 muestra una configuración típica usando una fuente precisa.

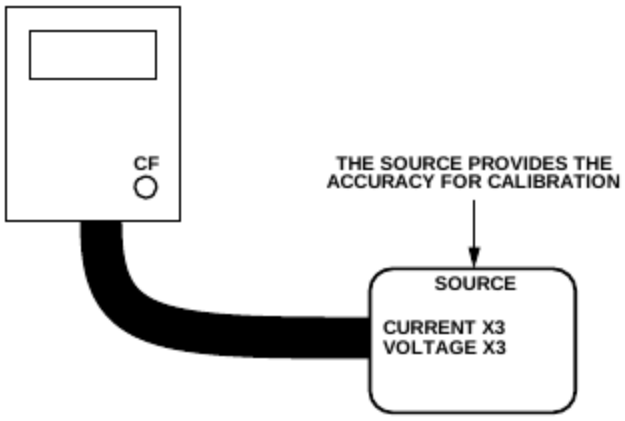


Figura 112: Conexión de una fuente precisa.

CONDICIONES DE ENTRADA PARA LA CALIBRACION

Como se muestra en la tabla 10, se requiere un máximo de tres pasos de calibración. Cada paso de calibración requiere una medición separada y un cálculo a realizar. Para permitir que se extraigan los errores de ganancia, fase y compensación por separado, normalmente se requieren tres conjuntos separados de condiciones de entrada. Estos se muestran en la tabla 11.

Etapas de calibración	Entrada de voltaje	Entrada de corriente	Factor de potencia
Ganancia.	Nominal	Nominal	0.5
Fase.	Nominal	Nominal	1
Desplazamiento.	Nominal	Mínima	1

Tabla 11: Condiciones de entrada estándar para la calibración.

Dónde: La tensión nominal es típicamente 120V o 240V. La corriente nominal es típicamente alrededor de 1/10 de la corriente máxima, tal como 10A. La corriente mínima es la corriente mínima especificada en el medidor mientras se mantiene dentro de la especificación de la medición para el ADE7880. Para acelerar el procedimiento de calibración y minimizar el número de condiciones de entrada, la calibración de ganancia también se puede realizar con un factor de potencia de 0.5. Esto permite utilizar un solo punto de calibración tanto para la calibración de ganancia como de fase. En muchos casos, esto reduce el procedimiento de calibración total a un solo punto, ya que no siempre se requiere la calibración por desplazamiento. La tabla 12 muestra las condiciones de calibración modificadas.

Etapas de calibración	Entrada de voltaje	Entrada de corriente	Factor de potencia
Ganancia.	Nominal	Nominal	0.5
Fase.	Nominal	Nominal	0.5
Desplazamiento.	Nominal	Mínima	1

Tabla 12: Condiciones de estrada estándar modificadas.

Al utilizar las condiciones de entrada que se muestran en la tabla 12, es importante que el factor de potencia utilizado sea 0.5 y que no varíe. Tenga en cuenta que se puede utilizar una carga inductiva o capacitiva.

AJUSTES DE REGISTRO REQUERIDOS

Antes de calibrar el ADE7880, es importante que se configure un conjunto de registros. Estos registros se enumeran en la tabla 13. Consulte la hoja de datos ADE7880 para obtener detalles sobre estos registros.

Dirección de los registros	Nombre de los registros	Descripción de los registros	Valores por defecto	Comentario
0xEA02	WTHR	Registro umbral de energía activa	0x03	Ecuación 26
0xEA03	VARTHR	Registro umbral de energía reactiva	0x03	Ecuación 37
0xEA04	VATHR	Registro umbral de energía aparente	0x03	Ecuación 44
0x4388	DICOEFF	Algoritmo integrador digital; sólo se requiere si se utilizan sensores di/dt	0xFFF8000	Solo para bobinas rowogski
0x439F	VLEVEL	Registro de umbral utilizado en el cálculo fundamental	0x38000	Ecuación 22
0xE60E	COMPMODE (SELFREQ)	Selección de 50 Hz o 60 Hz solo para medición fundamental	50Hz 0 60Hz 1	Requerido sólo para lecturas fundamentales.

Tabla 13: Registros requeridos para la calibración.

CALIBRACION DE LOS CANALES DE CORRIENTE Y VOLTAJE

Al calibrar utilizando la salida de pulso, el pin CFx debe estar configurado para emitir la medición y el canal que se está calibrando. Por ejemplo, al calibrar la energía activa en el canal A, configure CF1, CF2 o CF3 para que sea proporcional a la potencia activa en el canal A. Esto se logra ajustando el bit 0 al bit 8 del registro CFMODE (dirección 0xE610) así como el bit 0 al bit 8 del registro COMPMODE (Dirección 0xE60E). Pueden utilizarse CF1, CF2 o CF3. Para una calibración más rápida, se pueden realizar múltiples mediciones por canales diferentes en CF1, CF2 y CF3, simultáneamente, con hasta tres calibraciones realizadas en paralelo. Esto permite calibrar simultáneamente las tres fases. Es conveniente que coincidan las tres fases como parte de la calibración. Igualar las fases resulta en cálculos más fáciles porque un pulso en la salida CF tiene el mismo peso en cada fase. Por lo que el primer paso de calibración es el ajuste de los canales de entrada de corriente. Para ajustar las corrientes de fase A, B y C, aplique la misma corriente de entrada fija proveída por el instrumento de referencia a los transformadores de corriente de cada fase. Debido a que el medidor aún no ha sido calibrado, se recomienda que la amplitud de la señal aplicada esté entre la escala completa y 100:1. Las lecturas de las corrientes rms se pueden utilizar para determinar si hay algún error entre las corrientes de fase. Este error puede ser corregido utilizando los registros AIGAIN, BIGAIN y el registro CIGAIN respectivamente para cada fase. El instrumento de referencia que se utiliza para proveer de la corriente fija a los TC nos proporciona el valor de dicha corriente. Se utilizan los registros AIRMS, BIRMS y CIRMS para obtener las

mediciones reales rms de cada fase de corriente y comparar con el valor del instrumento de referencia. Si por lo menos una de las mediciones rms reales coinciden con el valor de corriente del instrumento de referencia entonces debe ajustar las otras dos fases usando los registros de ganancia. La siguiente ecuación describe cómo ajustar las lecturas.

$$xIGAIN = 2^{23} * \left(\frac{AIRMS}{xIRMS} - 1\right)$$

Donde:
x puede ser B o C según la fase y necesidad del ajuste.
Luego de haber ajustado las fases de corriente y obtener lecturas correctas. Aumente la corriente del instrumento de referencia en ±15% y repita los cálculos para las tres fases de corriente si los valores de medición reales rms son correctos la calibración finaliza. Para la calibración de los voltajes de fase se ejecuta un procedimiento similar al de la calibración de corriente. El instrumento de referencia proporciona un nivel de tensión alterna estable y se procede a conectar las fases y el neutro dependiendo de la configuración trifásica, luego los registros de ganancia para las fases de voltaje son AVGAIN, BVGAIN y CVGAIN respectivamente y los registros de las mediciones rms son AVRMS, BVRMS y CVRMS respectivamente. Para ajustar las lecturas se hace con la siguiente ecuación:

$$xVGAIN = 2^{23} * \left(\frac{AVRMS}{xVRMS} - 1\right)$$

Donde:
x puede ser B o C según la fase y necesidad del ajuste.

CALIBACION DE ENERGÍA

La salida de pulsos CFx se puede configurar de modo que cada pulso represente una fracción de kWh. Esta relación se conoce como la constante del medidor. Normalmente, las especificaciones de diseño requieren una constante de medidor particular para permitir que se verifique la precisión de los medidores de múltiples fabricantes. Las constantes típicas son 1600 imp/kWh, 3200 imp/kWh y 6400 imp/kWh. Si el diseño de un medidor que no requiere una constante específica del medidor, entonces en este caso se puede elegir un valor arbitrario. La salida de CFx se configura usando el divisor, CFxDEN. Este divisor se calcula sobre la base de la constante del medidor y la escala nominal en los canales de corriente y tensión. El CFx esperado se puede determinar bajo una carga dada. Para este trabajo de graduación el cálculo se hace con los siguientes valores nominales:

- Voltaje = 120Vrms.
- Corriente = 15Arms.
- Factor de potencia = 0.5
- Constante del medidor = 3200[imp/KWh]

Con esta carga, la frecuencia de salida de CFx se calcula como sigue:

$$CF_{ESPERADO} = \frac{CM\left[\frac{imp}{KWh}\right] * carga[KW]}{\frac{3600s}{h}}$$

Sustituyendo valores en la ecuación se tiene

$$CF_{ESPERADO} = \frac{3200\left[\frac{imp}{KWh}\right] * \frac{120V * 15A}{1000} * 0.5}{\frac{3600s}{h}}$$

$$CF_{ESPERADO} = 800E-3[Hz] \approx 0.8[Hz]$$

Se calcula un valor para el registro CFxDEN con el que se pueda obtener una frecuencia de 0.8Hz en las condiciones de carga dadas. La Figura 59 muestra el circuito de adquisición de señales de voltaje de donde se hace el siguiente análisis.

$$V_p = V_{ENTRADA\ MAXIMO} * \frac{1.5K\Omega}{1M\Omega + 1.5K\Omega}$$

$$V_p = 120 * \sqrt{2} * \frac{1.5K\Omega}{1M\Omega + 1.5K\Omega} = 254.18[mV]$$

$$V_{\%escala\ completa} = \frac{254.18E-3}{0.5} * 100\% = 50.83\%$$

Con una amplitud de canal de tensión de 120 V rms, la entrada funciona al 50.83% de la escala completa. La figura 57 muestra el circuito de adquisiciones de señales de corriente. Asumiendo una relación de transformación TC de 2000:1 y resistencias de burden de 3.5Ω, con una amplitud de canal de corriente de 15A rms, la entrada opera al 14.00% de la escala completa.

$$I_s = \frac{15A}{2000} = 7.5[mA]$$

$$V_{BURDEN} = I_s * R_{BURDEN} = (7.5E-3) * (6.6) = 49.5[mV]$$

$$I_{\%escala\ completa} = \frac{(49.5E-3) * \sqrt{2}}{0.5} * 100\% = 14.00\%$$

De la hoja de datos ADE7880, la salida máxima de CFx con entradas analógicas a escala completa es 68.818 kHz asumiendo que el registro este configurado así WTHR = 3. Cuando se aplica un factor de potencia de 0.5, esto se reduce a 34.409 kHz. Para obtener 0.8Hz con la entrada dada de 120V, 15A y PF = 0.5, el valor del registro CFxDEN debe ajustarse a 0xBF5, como se muestra en el cálculo a continuación:

$$CFxDEN = \frac{Frecuencia_{escala\ completa} * V_{\%escala\ completa} * I_{\%escala\ completa}}{CF_{ESPERADO}}$$

$$CFxDEN = 0xBF5$$

Al escribir 0xBF5 en el registro CFxDEN se establece la salida de CF alrededor de 0.8Hz para las condiciones descritas anteriormente. Esta configuración de CFxDEN ahora se puede utilizar en cada medidor. La calibración de ganancia proporciona una resolución más fina que debe hacerse en cada medidor para asegurar que los 0.8Hz se cumplen con precisión.

CALIBRACIÓN DE FASE

La calibración de fase es necesaria cuando se utiliza un transformador de corriente (TC) para eliminar cualquier cambio de fase introducido por el sensor. Los TC pueden añadir cambios de fase significativos que introducen grandes errores en factores de baja potencia. Si el uso de un sensor de corriente que no introduce un retardo de fase no es normalmente necesaria su calibración, ya que el ADE7880 está muy bien adaptado a la fase. La calibración de fase se realiza idealmente con una carga inductiva o capacitiva con un factor de potencia de 0.5. Si esta carga no está disponible, se puede elegir otro factor de potencia. Para obtener mejores resultados, el factor de potencia debe estar lo más cerca posible de 0.5. Para realizar la calibración de fase en un paso con una lectura. Las potencias activas y reactivas deben medirse simultáneamente. La siguiente ecuación describe cómo determinar el error de fase en grados.

$$error(°) = \frac{CF_{ACTIVO} \sin\varphi - CF_{REACTIVO} \cos\varphi}{CF_{REACTIVO} \sin\varphi + CF_{ACTIVO} \cos\varphi}$$

Dónde: φ se refiere al ángulo entre la tensión y la corriente en grados. Una vez determinado el error en grados, se puede utilizar la siguiente fórmula para determinar la compensación de fase requerida.

$$ResolucionFase = \left(\frac{360° * f}{1.024MHz}\right)$$

$$CompensacionFase = abs\left(\frac{error(°)}{ResolucionFase}\right)$$

Dónde: f se refiere a la frecuencia de línea. Tenga en cuenta que el formato del registro APHCAL es tal que si el valor del Error (°) es positivo, entonces se debe agregar un valor de 512d a la CompensacionFase calculado, antes de escribir en el registro APHCAL.

RESUMEN VALORES DE LOS REGISTROS DE CALIBRACION

REGISTRO	VALOR	REGISTRO	VALOR
AIGAIN	0x1A9C00	**COMPMODE**	0x41FF
BIGAIN	0x1C4C00	**CFMODE**	0x0CA0
CIGAIN	0x1A5D00	**CF1DEN**	0x0BF5

Tabla 14: Valores de los registros de calibración.

PRUEBAS DE EXACTITUD PARA UN MEDIDOR TRIFÁSICO

Para determinar la exactitud de los quipos de medición se hace atreves de medidores de referencia disponibles comercialmente. Uno de estos equipos es el medidor de la marca WECO Radian el medidor de pruebas portátil WE-20 con las siguientes especificaciones:
Precisión típica: ± 0.01% y en el caso más desfavorable la precisión es: ± 0.04%
Fuentes de corriente

- Salida de corriente: 0.10 - 30A por fase con resolución de 1mA. 50VA por fase.
- Precisión de corriente: ± 0.2%, ± 1 mA true RMS.
- Distorsión armónica de corriente: Menos del 1% THD en el peor de los casos (selección pura de ondas sinusoidales).

Fuentes de potencial

- Potencial de salida: 30-600V AC con resolución 10mV, 35VA por fase.
- Precisión de las fuentes de potencial: ± 0.2%, ± 10 mV true RMS.
- Distorsión armónica de las fuentes de potencial: Menos del 1% de THD en el peor de los casos (selección pura de ondas sinusoidales).

Control de fase: 0° - 359.95° en pasos 0.05°
Precisión de fase: ± 0.10°
Frecuencia de ensayo: 45 - 65Hz en pasos de 0.001Hz
Precisión de la frecuencia de ensayo: 50ppm f
Rango de entrada de alimentación auxiliar: 90-300V AC auto-rango.
Rango de frecuencia de entrada: 45 - 65Hz (fundamental)
Peso del sistema base: 44 libras.
Dimensiones de la caja: 24.7pul L 24pul W 12pul D
Temperatura:
0°C a 55°C (32°F a 131 °F) operación
-20°C a 70°C (-4°F a 158°F) almacenamiento
Humedad: 0%-95% sin condensación

PROCEDIMIENTO PARA LLEVAR ACABO LAS PRUEBAS DE EXACTITUD

Con la parte de calibración desarrollada satisfactoriamente, se procede a la configuración del medidor de referencia para realizar las pruebas de exactitud. Los valores nominales son:
Kh = CM = 3200[imp/KWh]
Voltaje nominal = 120Vrms
Corriente de prueba carga alta 15Arms
Corriente de prueba carga baja 1.5Arms
Forma del medidor: 9s, 12s, 16s…etc
Factor de potencia de prueba: carga alta y carga baja = 1.
Prueba con el factor de potencia a 0.5.

Selección del tipo de sensor óptico a utilizar.
Estos son los valores de prueba para el medidor desarrollado en este trabajo de graduación.
A continuación se hacen las conexiones desde las fuentes de potencial y corriente del medidor de referencia hacia el medidor que se desea probar. Una vez realizadas las conexiones se procede a ejecutar las pruebas de exactitud. En las figuras 113, 114 y 115 se trata de ilustrar a manera de ejemplo este proceso.

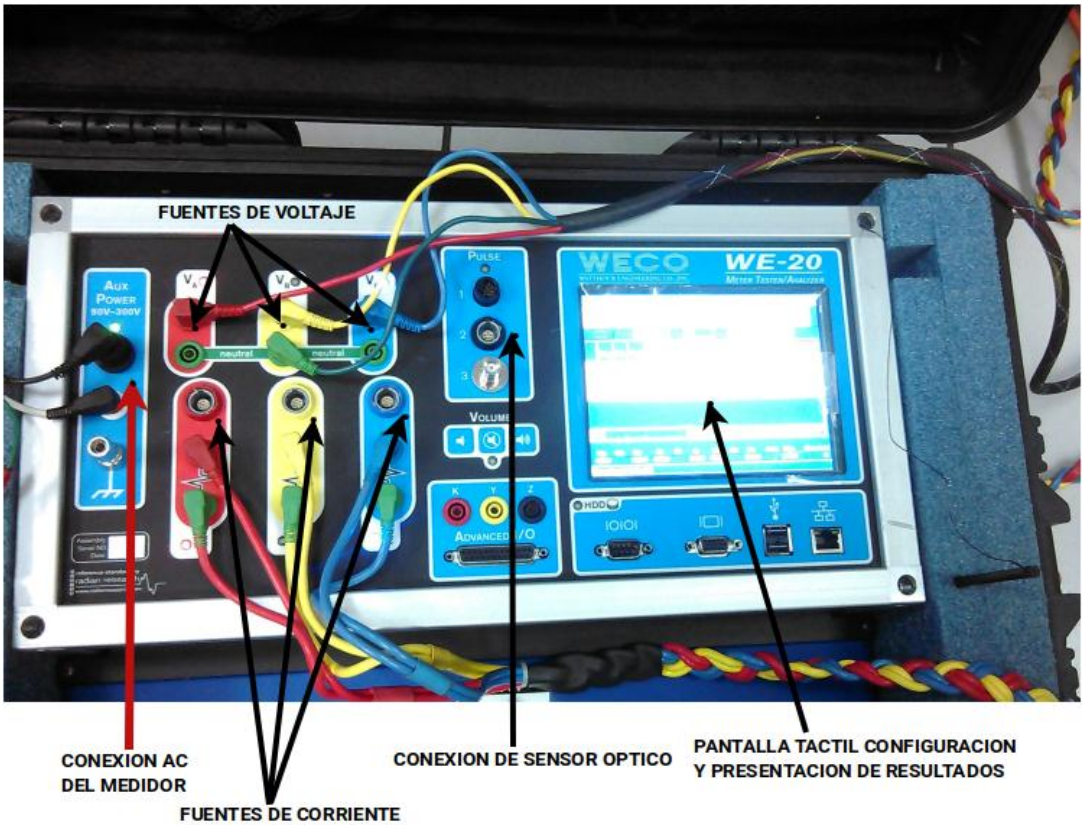


Figura 113: medidor de referencia weco radian WE-20.

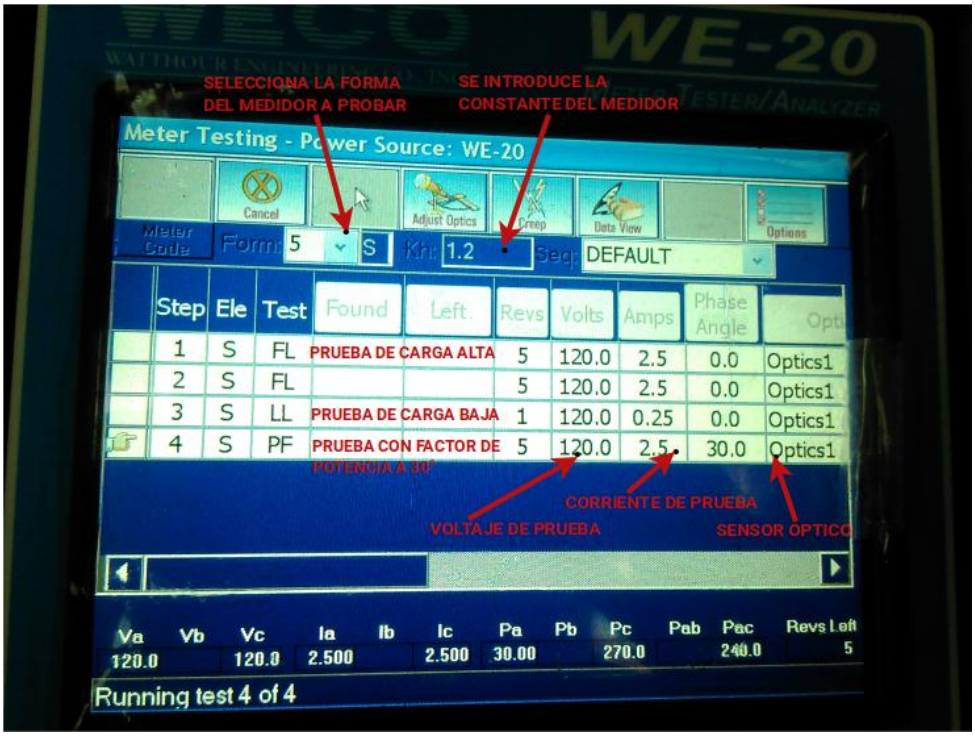


Figura 114: Parámetros de configuración para realizar las pruebas de exactitud.

Figura 115: Resultados de las pruebas de exactitud para un medidor trifásico.

Anexo I. Presupuesto económico invertido en el prototipo y Manual de usuario

CANTIDAD	DESCRIPCION	PRECIO UNITARIO	SUBTOTAL	ENVIO
12	RESISTENCIA 1.5K	0.3	3.6	LOCAL
8	CAPACITORES 22n	0.5	4	LOCAL
4	CAPACITORES 22n	0.85	3.4	LOCAL
16	RESISTENCIA 1.5	0.1	1.6	LOCAL
8	RESISTENCIA 2.0	0.1	0.8	LOCAL
4	RESISTENCIA 1M	0.3	1.2	LOCAL
3	CAPACITORES 4.7u	0.28	0.84	LOCAL
1	CAPACITORES 10u	0.34	0.34	LOCAL
2	CAPACITORES 0.22u	0.25	0.5	LOCAL
2	CAPACITORES 0.1u	0.5	1	LOCAL
2	CAPACITORES 22p	0.2	0.4	LOCAL
20	XTAL 16.384MHz	44.01 + 30.4	74.41	DHL + ADUANA SV
4	SCT013 100A/50mA	5 + 3.99	23.95	CORREOS SV
4	JACK conectores para los TC	0.4	1.6	LOCAL
4	TENAZAS	----	6.2	LOCAL
4	Mts cable sae 18	----	1.04	LOCAL
1	CAUTIN	5	5	LOCAL
5	Mts estaño	3	3	LOCAL
1	Pasta para soldar	3.5	3.5	LOCAL
20	Mts filamento PLA	7.5	7.5	CORREOS SV
20	Mts filamento ABS	7.5	7.5	CORREOS SV
5	LED	0.5	2.5	LOCAL
1	RTC	5.5	5.5	CORREOS SV
1	ADE7880	15	15	CORREOS SV
1	PCB MEDIDOR	30	30	FEDEX + ADUANA SV
1	PCB PARA SEÑALES	77.1 + 23.85	100.95	DHL + ADUANA SV
1	Raspberry pi 4 1GB	97.79	97.79	FEDEX + ADUANA SV
1	Memoria microSD 32GB	8	8	LOCAL
4	Pines header socket	0.55	2.2	LOCAL
1	Jumper wire 50 unidades	2.5	2.5	LOCAL
	TOTAL		**$415.82**	

Tabla 15: Presupuesto para el equipo de medición

Se debe contar con el equipo de seguridad antes de llevar a cabo la conexión del medidor al sistema eléctrico trifásico. Por lo tanto debe retirarse del cuerpo cualquier objeto metálico (cadenas, pulseras, relojes, anillos, etc…) y contar con el siguiente equipo básico de protección:

- Guantes para electricista baja tensión o los de clase 00 según norma ASTM.
- Casco con protector facial de preferencia.
- Camisa para electricista antiflama o según normas NFPA 70E y NFPA 2112.
- Botas del tipo dieléctricas.

Habiendo cumplido con lo antes expuesto se procede a efectuar las conexiones.
Conectar el medidor aun sistema eléctrico trifásico en configuración estrella.
Para ello se realizan las conexiones que se muestran en la figura 116.

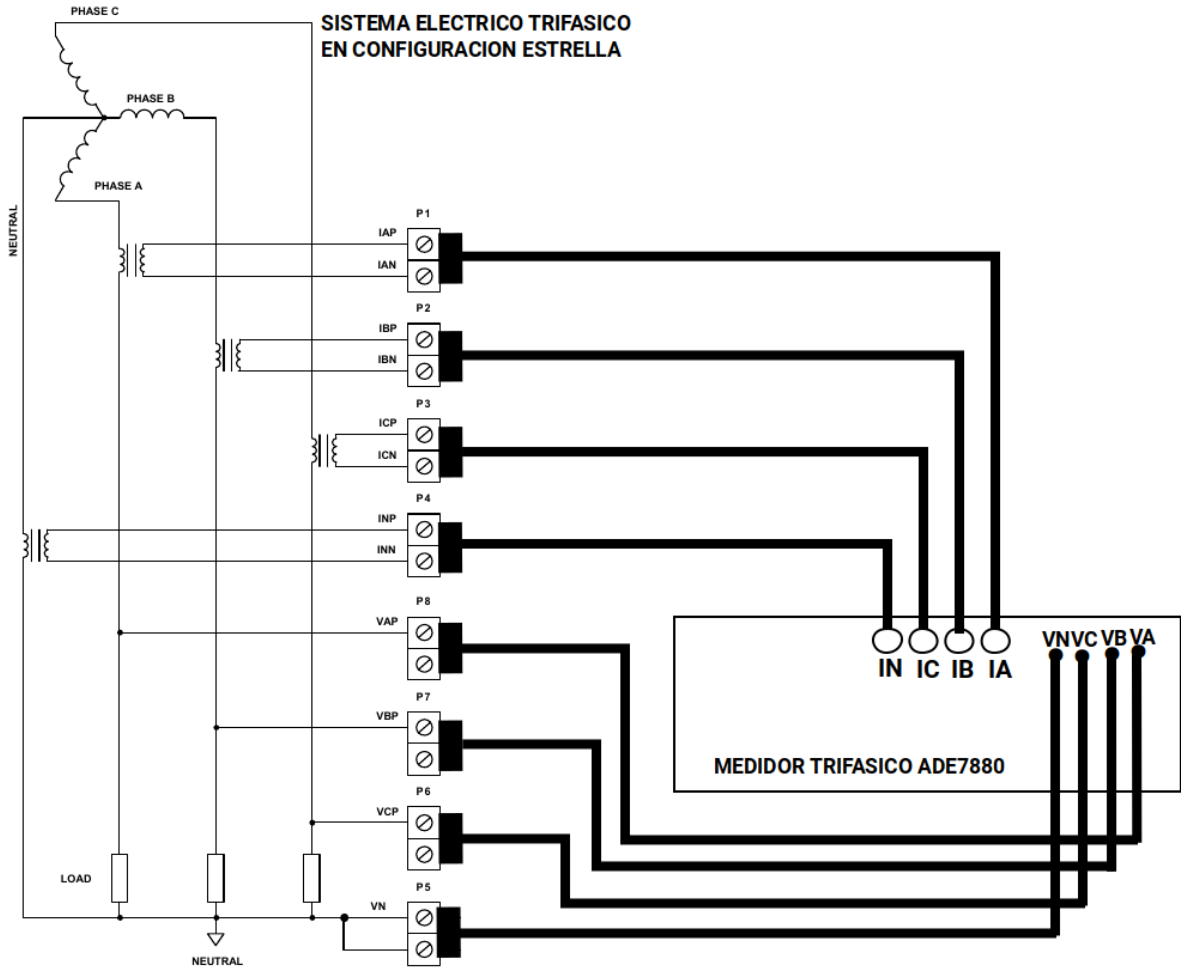


Figura 116: Conexiones para un sistema eléctrico trifásico en configuración estrella.

Se procede a identificar las líneas de conexión de voltaje y corriente en el medidor en el orden que se ha mostrado en la figura 116. Para conectar los transformadores de corriente se identifica la fase en el tablero general y la misma fase en las entradas de corriente del medidor, luego se abre el transformador se coloca en el centro del cable de la fase de interés y se cierra el transformador asegurándose que el cierre quede fijo. Se desarrolla el mismo procedimiento para las demás fases de corriente. En cuanto a las conexiones de voltaje se debe ser más precavido ya que son tenazas metálicas las que se usan para conectar en los bornes del tablero general. Se identifica la fase a conectar tanto en el medidor como en el tablero general en ambas debe ser la misma a continuación se abre la tenaza y se coloca fijamente sobre la bornera de la fase de interés. De la misma forma se realizan las conexiones de voltaje de las demás fases. Después de haber conectado el medidor al sistema eléctrico trifásico se configura el tipo de conexión trifásica por software como se muestra en la figura 117.

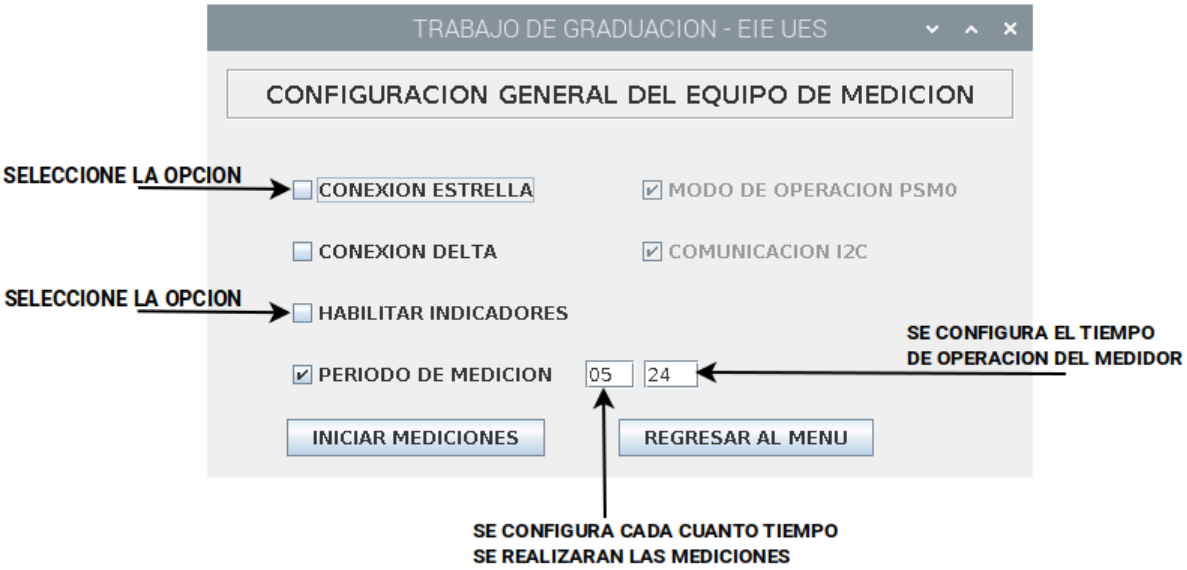


Figura 117: Configuración por software del sistema eléctrico trifásico en estrella.

Al terminar la configuración se presiona el botón iniciar mediciones y luego el botón regresar al menú y con esto el equipo de medición está configurado y realizando mediciones de acuerdo a lo configurado.

Conectar el medidor aun sistema eléctrico trifásico en configuración delta 3 líneas.
Para ello se realizan las conexiones que se muestran en la figura 118. Se procede a identificar las líneas de conexión de voltaje y corriente en el medidor en el orden que se ha mostrado en la figura 118. Para conectar los transformadores de corriente se identifica la fase en el tablero general y la misma fase en las entradas de corriente del medidor, luego se abre el transformador se coloca en el centro del cable de la fase de interés y se cierra el transformador asegurándose que el cierre quede fijo. Se desarrolla el mismo procedimiento para las demás fases de corriente. En cuanto a las conexiones de voltaje se debe ser más precavido ya que son tenazas metálicas las que se usan para conectar en los bornes del tablero general. Se

identifica la fase a conectar tanto en el medidor como en el tablero general en ambas debe ser la misma a continuación se abre la tenaza y se coloca fijamente sobre la bornera de la fase de interés. De la misma forma se realizan las conexiones de voltaje de las demás fases. Después de haber conectado el medidor al sistema eléctrico trifásico se configura el tipo de conexión trifásica por software como se muestra en la figura 119.

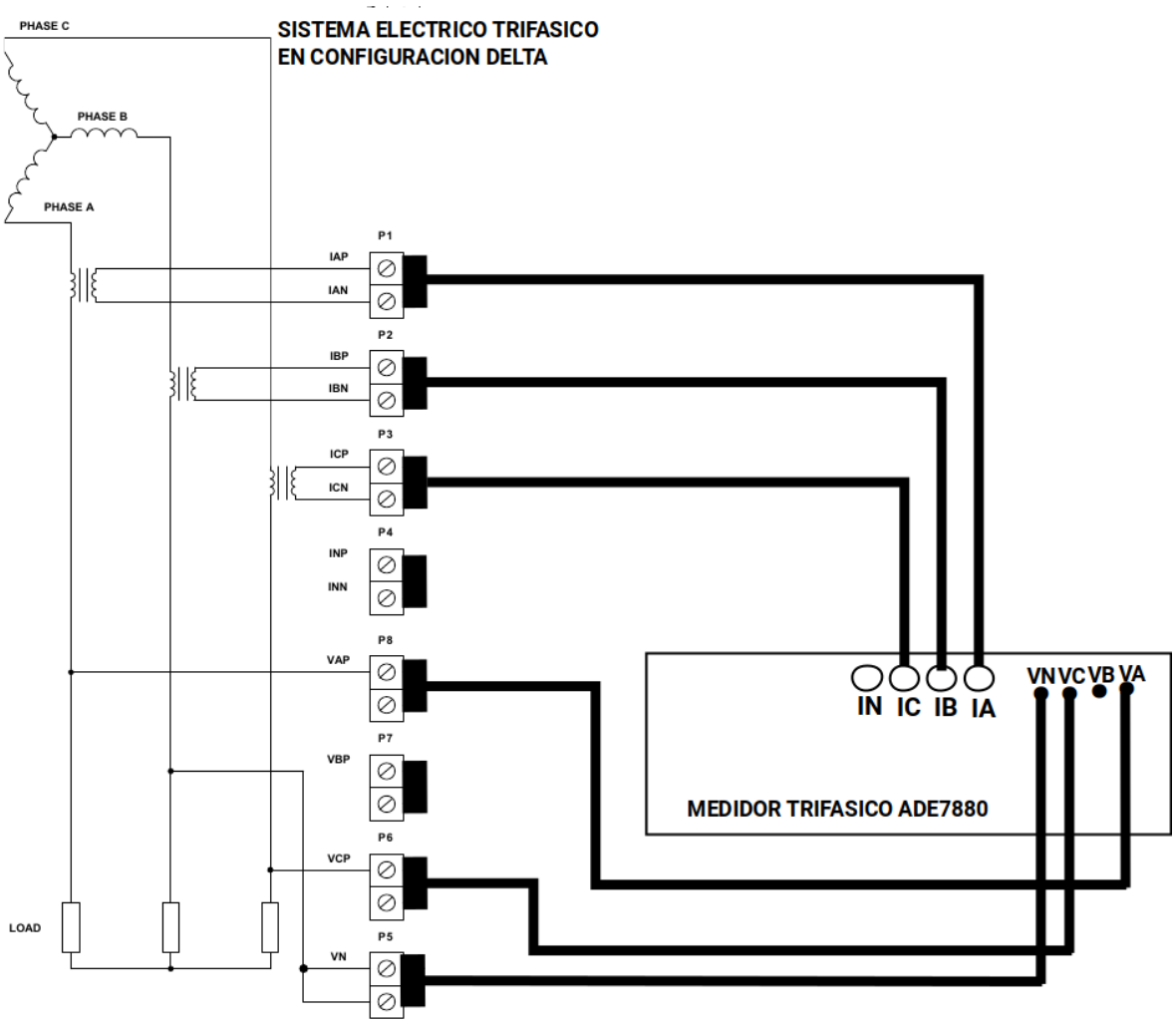


Figura 118: Conexiones para un sistema eléctrico trifásico en configuración delta.

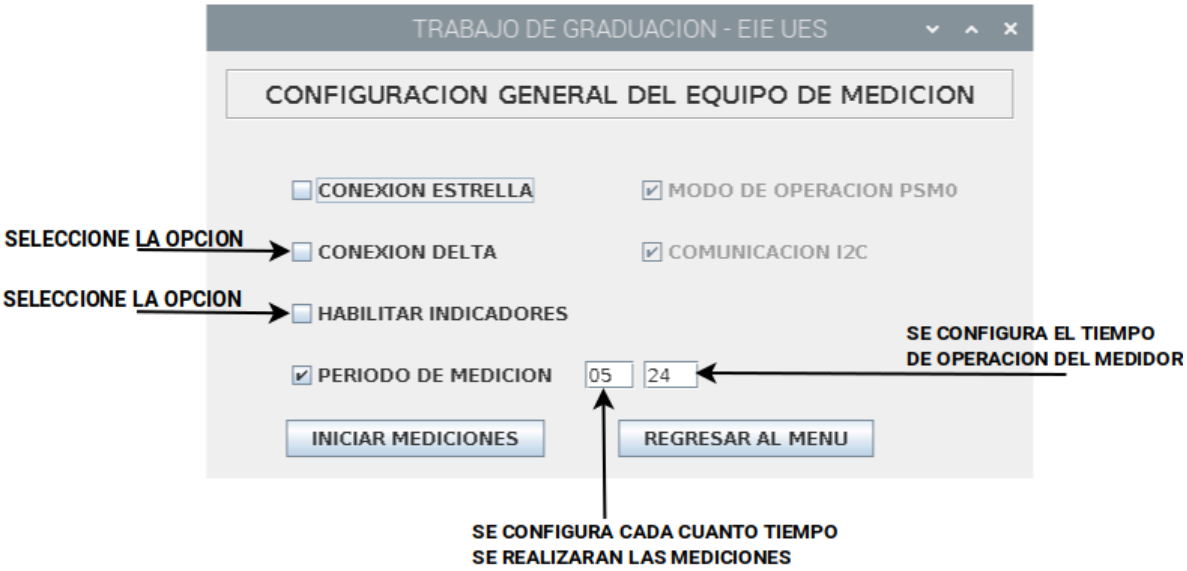


Figura 119: Configuración por software del sistema eléctrico trifásico en delta.

Para habilitar o deshabilitar la comunicación remota se efectúa como se muestra en la figura 120. Luego se presiona el botón regresar al menú.

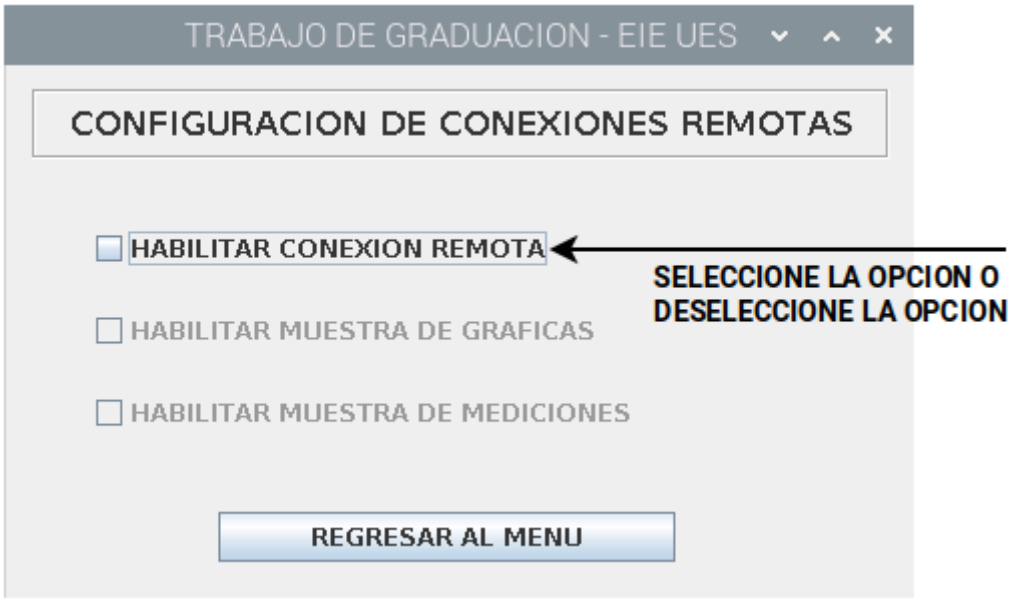


Figura 120: Configuración de la conexión remota.