UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

Equipo concentrador de vatímetros programables para uso en el sector doméstico e industrial

PRESENTADO POR:

JULIO OTONIEL MURCIA FLORES

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, NOVIEMBRE DE 2014

UNIVERSIDAD DE EL SALVADOR

RECTOR :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALETA DE AMAYA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERIA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

EQUIPO CONCENTRADOR DE VATÍMETROS PROGRAMABLES PARA USO EN EL SECTOR DOMÉSTICO E INDUSTRIAL

Presentado por :

JULIO OTONIEL MURCIA FLORES

Trabajo de Graduación Aprobado por:

Docente Director :

ING. HUGO MIGUEL COLATO RODRÍGUEZ

San Salvador, noviembre de 2014

Trabajo de Graduación Aprobado por:

Docente Director :

Ing. Hugo Miguel Colato Rodríguez

AGRADECIMIENTOS

Agradezco a Dios Todopoderoso por la ayuda que me ha brindado hasta este día, por mi fe en él he podido superar los momentos más difíciles en mi vida y le agradezco todas esas dificultades que han forjado lo que soy ahora, porque no hay un manjar más delicioso que aquel que se come después de una dura jornada.

Agradezco a mi padre y madre el apoyo, amor y cuidados que me han brindado, aun en las situaciones difíciles que hemos enfrentado como familia, ellos siempre han hecho esfuerzos cuasi sobrehumanos para solventar las necesidades de mi hogar, también les agradezco por todas las buenas costumbres y enseñanzas que me han heredado, esas que no se aprenden en la escuela, el colegio o en la universidad.

Agradezco a mi abuela y a todos aquellos que han puesto palabras en sus oraciones pidiendo que la ayuda y bendición de Dios estén siempre presentes en mi vida, pero lamento la ausencia de mis abuelos que me brindaron su apoyo y deseaban ver culminada mi carrera.

Agradezco a mi hermano Nehemías por toda la ayuda que me ha dado y aunque hemos recorrido caminos diferentes compartimos los mismos sueños e ideales.

Agradezco a todo el personal de la Universidad de El Salvador y en específico a la Escuela de Ingeniería Eléctrica que me ayudaron a adquirir el conocimiento otorgado por los grandes personajes de la ciencia y tecnología de la historia, además de las alegrías, tristezas, amistades y enemistades que siempre tendrán un lugar especial en mi corazón y en mis recuerdos.

JULIO OTONIEL MURCIA FLORES.

PREFACIO

Con el propósito de diseñar y ensamblar un equipo prototipo medidor de potencia programable aplicable tanto en el sector industrial como en el doméstico. Se ha incluido en el presente documento una breve recopilación bibliográfica sobre los diferentes sistemas de comunicación y dispositivos electrónicos que conforman la base teórica del presente trabajo.

Hoy en día, los medidores de estado sólido son sinónimo de precisión, exactitud y automatización en el campo de las mediciones eléctricas; debido a que están diseñados con componentes electrónicos que no son susceptibles a desgastes mecánicos, lo que permite mayor autonomía, datos más confiables y capacidad de comunicarse con una computadora.

Estos sistemas electrónicos de medición permiten obtener los mismos parámetros eléctricos que los medidores electromecánicos, y poseen nuevas características como memoria de almacenamiento, ajustes de calibración, así como más opciones de configuración.

Estos dispositivos electrónicos son capaces de medir la potencia reactiva que no se convierte en trabajo útil permitiendo a la compañía distribuidora facturar dichas perdidas; y al usuario corregir el consumo de potencia reactiva para evitar penalizaciones.

RESUMEN DE TRABAJO.

Se diseña e implementa un equipo eléctrico que mide la potencia, voltaje y corriente, de un sistema trifásico utilizando un circuito integrado especializado producido por la empresa Analog Devices.

El equipo consta de módulos que miden los parámetros previamente mencionados, cuyos datos son transmitidos a un módulo central (concentrador) utilizando el protocolo MODBUS/CAN, para este módulo central se empleará el sistema SIPROE[1], desarrollado en una tesis previa, al cual se le dotará de la funcionalidad de medición de energía eléctrica.

Se le dará al usuario final la capacidad de limitar el consumo de energía eléctrica máxima, lo que conlleva que el sistema al detectar que se ha rebasado el máximo permisible enviará una señal a unos actuadores que cortaran el suministro de energía eléctrica a la carga.

Por medio del sistema SIPROE se podrá tener acceso a los datos almacenados en un servidor vía Ethernet.

En el capítulo I, se desarrollan las bases teóricas necesarias para comprender la filosofía de funcionamiento de cada una de las partes que componen el equipo eléctrico implementado.

En capítulo II, se detalla cada parte o componente del diseño del hardware utilizado para implementar el prototipo del módulo medidor de energía eléctrica, que se agregará al sistema SIPROE.

En el capítulo III, Se describe cada una de las partes del desarrollo software que ofrece una Interface hombre maquina (HMI) amigable para la comunicación con los diferentes controladores y la administración de dispositivos según jerarquía de roles. Además de describir la interacción del usuario a través del aplicativo web tanto con la base de datos como el aplicativo intermedio el cual provee la comunicación con el hardware del sistema.

[1] SIPROE: **S**istema **I**nteligente **pr**ogramable **O**ptimizador de **E**nergía Eléctrica.

TABLA DE CONTENIDO.

INDICE DE FIGURAS.

INDICE DE TABLAS.

OBJETIVOS.

OBJETIVO GENERAL.

Diseño e implementación de un equipo eléctrico tal que se pueda proveer una solución de ahorro energético para el sector doméstico o empresarial.

OBJETIVOS ESPECÍFICOS.

- Desarrollar módulos hardware para hacer mediciones de los parámetros eléctricos: corriente, voltaje y potencia.

- Poder almacenar las mediciones en una base de datos y poder extraer los datos a través de una red Ethernet.

- Utilizar equipo desarrollado en proyectos anteriores.

CAPITULO I.

MARCO TEÓRICO.

1.1. SENSORES DE CORRIENTE.

Un sensor es un dispositivo capaz de medir magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Estas variables de instrumentación son muy variadas y están presentes en todo tipo de fenómenos físicos y químicos, dentro de estas se puede mencionar la temperatura, la presión, la posición, la humedad, la fuerza, la intensidad luminosa, la aceleración, el caudal, el pH, la corriente, el voltaje, etc.

Este tipo de instrumentos tienen múltiples aplicaciones, son mayoritariamente usados en las empresas para poder medir y controlar todas las variables de instrumentación asociadas a los procesos que allí se lleven a cabo. Son muy utilizados en la industria automotriz, en la industria aeroespacial, en la industria manufacturera, medicina, robótica, etc.

1.1.1. HISTORIA DE LOS SENSORES DE CORRIENTE.

En los comienzos de la electrónica con la ausencia de los sensores, esta área era incapaz de percibir variables como temperatura, velocidad, humedad y corriente, por lo que se podría decir que el trabajo era completamente a ciegas, sin control alguno para un trabajo seguro y eficiente. Hace ya más de 80 años se ha comenzado con la búsqueda de sensores artificiales que ayuden al registro de los parámetros. Sin embargo, es en los últimos 20 años cuando los investigadores y la tecnología han

desarrollado un sin número de sensores físicos, electro-químicos y ópticos. Es así como con la aparición de los sensores le dio sentidos a la tecnología. Con ellos, las máquinas comenzaban a recibir información que, una vez procesada, permite generar la respuesta más adecuada en un momento concreto, ya sea abriendo una puerta, haciendo saltar una alarma, alertando de un movimiento sísmico, entre otros muchos ejemplos. Todo ello sin necesidad de ser activadas por la mano del hombre. Dentro de ellos tenemos los sensores de corriente los cuales destacan por detectar la corriente tanto alterna como continua. Podemos decir así que la información que aportan ha cobrado un valor extraordinario en todos los ámbitos de la actividad humana.

1.1.2. TIPOS DE SENSORES DE CORRIENTE.

Son 4 los tipos de sensores de corriente. Estos son Transformador de corriente, Bobina Rogowski, Shunt de corriente de baja resistencia y Sensor de efecto Hall los cuales detallaremos a continuación.

1.1.3. TRANSFORMADOR DE CORRIENTE.

Un transformador es un artefacto electromagnético, que permite disminuir o aumentar la tensión de un circuito eléctrico, o pasar de corriente alterna a corriente continua Los transformadores son dispositivos basados en el fenómeno de la inducción electromagnética y están constituidos, en su forma más simple, por dos bobinas devanadas sobre un núcleo cerrado. En un transformador ideal, la potencia que ingresa al equipo es igual a la de salida. Pero en la realidad, siempre existirá una pequeña pérdida, dependiendo del tamaño o diseño, etc.

Entre 1884 y 1885, los ingenieros húngaros Zipernowsky, Bláthy y Deri de la compañía Ganz crearon en Budapest el modelo "ZBD" de transformador de corriente alterna, basado en un diseño de Gaulard y Gibbs (Gaulard y Gibbs sólo diseñaron un

modelo de núcleo abierto). Descubrieron la fórmula matemática de los transformadores.

Los transformadores de corriente cumplen un rol fundamental en la vida cotidiana de las personas en general y de las empresas que hacen de la utilización de estos dispositivos una parte fundamental del funcionamiento de la amplia gama de industrias que existen en nuestro país y en el mundo.

1.1.3.1. CARACTERÍSTICAS.

Los transformadores tienen la capacidad de transformar el voltaje y la corriente a niveles más altos o más bajos. Está construido superponiendo numerosas chapas de aleación acero–silicio, con el fin de reducir las pérdidas por histéresis magnética y aumentar la resistividad del acero.

No crean la energía a partir de la nada; por lo tanto, si un transformador aumenta el voltaje de una señal, reduce su corriente; y si reduce el voltaje de la señal, eleva la corriente. En otras palabras, la energía que fluye a través de un transformador, no puede ser superior a la energía que haya entrado en él.

1.1.3.2. APLICACIONES.

Para transportar corriente, los transformadores permiten llevar tensiones elevadas, con baja intensidad para minimizar la pérdida de potencia ocasionada por la resistencia de los conductores. Para circuitos electrónicos como los de radio, podemos utilizar transformadores de impedancia.

En el uso cotidiano, la mayoría de los artefactos eléctricos que utilizamos, requieren de un *transformador* para ajustar la corriente de la red a los requerimientos

de funcionamiento. Por ejemplo, los monitores de las computadoras, los aparatos reproductores de CDs, los teléfonos celulares, etc.

1.1.3.3. COMPONENTES.

Si bien es cierta la cantidad de transformadores de corriente disponibles hoy en día en el mercado son de una gran variedad y aplicaciones, todos presentan una columna vertebral que es invariante. Puede que cambien en su forma de construcción del transformador en sí, pero todos cuentan con un núcleo magnético elaborados preferentemente con una aleación de hierro – silicio (hierro preferentemente), el cual constituye el circuito magnético cuya función principal es la de conducir el flujo magnético. Luego nos vamos a encontrar con bobinados, los cuales están ensamblados alrededor del núcleo magnético y sujetado a una estructura de soporte. Se encuentran dos bobinas, la primaria la cual tiene por función crear un campo magnético con una pérdida de energía muy pequeña y una bobina secundaria que aprovecha el flujo magnético para producir una fuerza electromotriz. Estas bobinas pueden ser de alambre delgado, grueso o barra y los materiales comúnmente utilizados son el cobre y aluminio. Un punto importante de mencionar es la razón de transformación entre la bobina primaria y secundaria, que depende del número de vueltas que tenga cada uno, por lo que puede ser elevador o reductor dependiendo del número de espiras de cada bobina. Todo esto se instala en una caja la cual puede estar fabricada en aluminio para la corrosión o en plástico para el aislamiento.

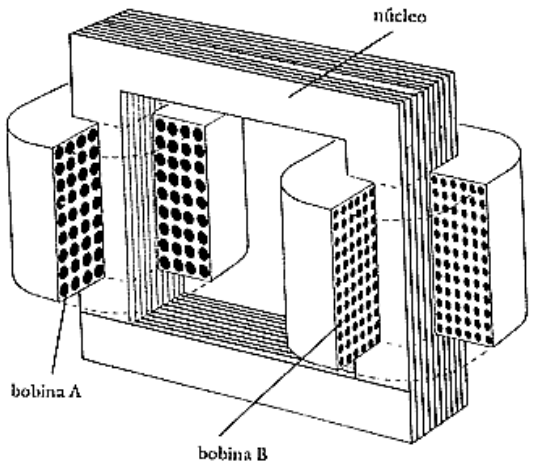

Figura 1.1: Representación esquemática del transformador.
Fuente [1]

1.1.4. BOBINA ROGOWSKI.

La bobina Rogowski, llamada así en honor a su inventor Walter Rogowski, es un dispositivo electrónico, usado como transductor para medir corriente alterna (AC) o pulsos rápidos de corriente. Esta consiste en una bobina uniformemente arrollada en un núcleo de material no magnético de sección transversal constante, distribuido en forma de lazo cerrado. La forma más simple es la de un toroide circular cerrado y rígido, o abierto y flexible para que pueda cerrarse sobre sí mismo y así facilitar su montaje alrededor de un conductor por el que circula la corriente a evaluar. La función de esta bobina es aplicar la Ley de Ampere, la cual dice que la corriente que circula por un conductor es proporcional a la integral de circulación de la intensidad de campo magnético alrededor de un camino cerrado que rodea a dicho conductor.

La idea anterior puede expresarse también diciendo que la bobina Rogowski, cuyo principio de funcionamiento se conoce desde 1912, se basa en medir los cambios del campo magnético que se produce alrededor de un hilo portador de corriente para producir una señal de voltaje, la cual es proporcional a la derivada de la corriente (di/dt), para lo que se necesita un integrador que convierta apropiadamente la señal.

La tarea de crear un integrador que fuera estable y exacto durante la larga vida de un medidor había desalentado a los estudiosos de este rubro. Sin embargo, la reciente implementación digital del integrador tiene la promesa de convertir esta tecnología en una realidad para los medidores eléctricos (Ver Figura 1.2).

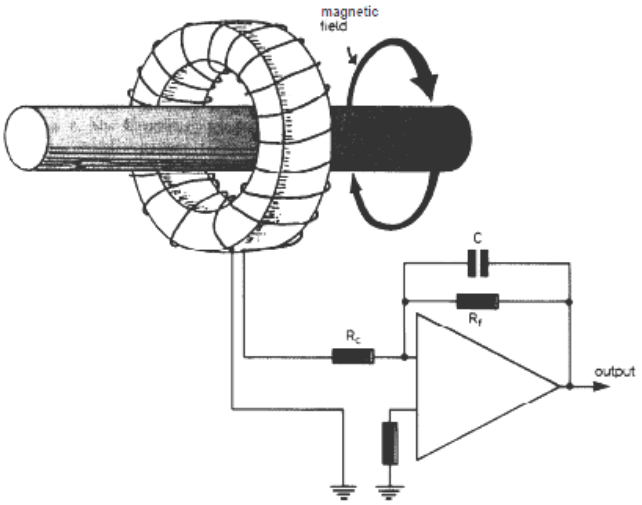

Figura 1.2: Esquema de un Bobina Rogowski con integrador.
Fuente [2]

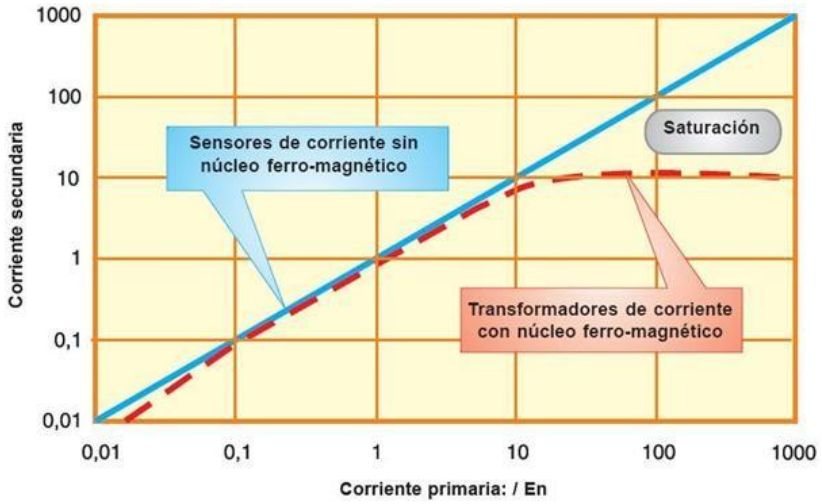

Figura 1.3: Comportamiento de linealidad, efecto Rowoski.
Fuente [1]

Dentro de las grandes ventajas de utilizar la bobina Rogowski es que al poseer núcleo de aire no presenta histéresis, saturación o problemas de no linealidad (ver figura 1.3). Además posee una elevada capacidad para manejar altas corrientes, siendo el límite superior teórico de la bobina el voltaje de ruptura del mismo aire. Por último se destaca que posee un bajo costo con respecto a otras tecnologías de sensores de corriente.

Algunas aplicaciones de este tipo de sensores los podemos ver en los reconectadores de media detención, ya que gracias a su gran precisión y características especiales de esta tecnología es posible mejorar considerablemente las prestaciones de protección, ya que permite medir las corrientes de fase y residual con algunas ventajas claves sobre las bobinas con núcleos de aceros.

1.1.5. SHUNT DE CORRIENTE DE BAJA RESISTENCIA.

De los distintos tipos de sensores, los Shunt de corriente son los de menor valor, además de ofrecer una lectura sencilla y con una buena precisión. Este tipo de sensor es la solución más utilizada (popular) para la medición de corriente.

Cuando se realizan las mediciones de corriente con alta precisión, es necesario tener en cuenta la inductancia del Shunt (relación entre el flujo magnético y la intensidad de corriente eléctrica, generalmente positiva) y aunque sólo a frecuencias relativamente altas afecta la magnitud de la impedancia (medida que establece la relación (cociente) entre la tensión y la intensidad de corriente), ya que su efecto es suficiente para provocar un error a un bajo factor de potencia. Por lo tanto es importante seleccionar una resistencia "Shunt" apropiada de sensor de corriente que debe tener un valor muy bajo de resistencia para minimizar la disipación de potencia, un valor bajo de inductancia y una tolerancia razonablemente pequeña para mantener una precisión global en el circuito.

Por ejemplo, los Shunt que se utilizan en instrumentos portátiles o de laboratorio están preparados para una caída de tensión de 60 mV. Para instrumentos de tablero se emplean los Shunt con caídas normalizadas de: 30, 45, 60, 100, 120, 150, 300 mV. Los Shunt se clasifican, según su exactitud, en cinco clases: 0.05, 0.1, 0.2, 0.5 y 1%. Cuando el instrumento (mili voltímetro) se conecta con el Shunt mediante cables, su calibración se efectúa en conjunto con los cables. Los métodos de medición se dividen en cuatro grupos principales: métodos voltiamperimétricos (técnicos), métodos de cero (puentes), métodos de deflexión y métodos de compensación. Cada uno de estos métodos tiene su campo de aplicación que se rige por la precisión requerida, por el alcance de la magnitud medida y por la disponibilidad del equipo.

Aunque es posible adquirir resistencias "Shunt" de sensor de corriente a fabricantes como: IRC, Dale, Ultronix, Isotek, y K-tronics, que son sólo algunos de los proveedores que fabrican resistencias apropiadas para aplicaciones de sensor de corriente, es también posible hacer una resistencia sensor utilizando diversos materiales interesantes como el cobre o la manganina[2].

En la figura 1.4 se muestran varios tipos de Shunt, que por lo general poseen cuatro bornes, que es el nombre dado en electricidad a cada uno de los terminales de metal en que suelen terminar algunas máquinas y aparatos eléctricos, y que se emplean para su conexión a los hilos conductores, como también lo muestra la figura 5, que presenta una resistencia Shunt sin su cuerpo protector. Esta geometría permite evitar errores causados por la resistencia de contactos.

[2] Es una aleación de 82-84% de cobre, 12-15% de manganeso y 2-4% de níquel. Es una aleación con un muy bajo coeficiente de temperatura y es muy utilizado en galgas extensiométricas (o extensómetro es un sensor, para medir la deformación, presión, carga, torque, posición, entre otras cosas, que está basado en el efecto piezorresistivo, el cual es la propiedad que tienen ciertos materiales de cambiar el valor nominal de su resistencia cuando se le someten a ciertos esfuerzos y se deforman en dirección de los ejes mecánicos) y resistores de alta estabilidad.

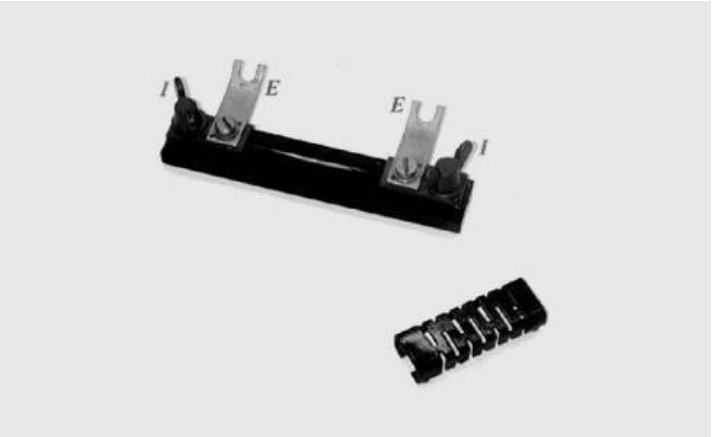

Figura 1.4: Distintos tipos de Shunt.
Fuente [1]

1.1.6. EL SENSOR DE EFECTO HALL.

Para ilustrarse este tipo de sensores es necesario conocer primero el Efecto Hall, el cual es una consecuencia de la fuerza que se ejerce sobre una carga eléctrica en movimiento cuando se encuentra sometida a la acción de un campo eléctrico y un campo magnético. En 1879 el físico E. Hall descubrió que cuando un conductor sobre el que circulaba corriente era colocado en un campo magnético de dirección perpendicular a la misma, podía medirse una pequeña diferencia de potencial en la dirección perpendicular a la corriente y al campo.

El sensor de efecto Hall sirve para la medición de campos magnéticos o corrientes, o para la determinación de la posición.

Su función sigue unos pasos en donde si fluye corriente por un sensor Hall y se aproxima a un campo magnético que fluye en dirección vertical al sensor, entonces el sensor crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente. Si se conoce el valor de la corriente, entonces se puede calcular la fuerza del campo magnético; si se crea el campo magnético por

medio de corriente que circula por una bobina o un conductor, entonces se puede medir el valor de la corriente en el conductor o bobina.

Si tanto la fuerza del campo magnético como la corriente son conocidas, entonces se puede usar el sensor Hall como detector de metales.

Existen dos tipos principales de sensores de Efecto Hall, anillo abierto (open-loop) y anillo cerrado (closed-loop). El segundo ofrece mejor precisión y rangos dinámicos más amplios pero a un costo mayor, y la mayoría de los sensores de Efecto Hall que se encuentran en medidores de energía usan el diseño anillo abierto para lograr costos más bajos. El sensor de Efecto Hall tiene una excelente respuesta a la frecuencia y está capacitado para medir corrientes muy altas.

La señal obtenida del sensor Hall puede ser procesada para dar una señal digital o analógica. De modo que cuando desea obtener una salida los sensores se denominan interruptores Hall. Y cuando se requiere que la salida sea proporcional a la señal que se desea medir, se denominan sensores Hall de tipo lineal. Estos últimos son los empleados para la medida de corrientes y cubren un rango que se extiende desde pocos mA hasta cientos de mA. En la figura 1.5 se muestran algunos sensores de corriente que emplean el efecto Hall.

Algunas aplicaciones para los sensores Hall los podemos encontrar en la industria del automóvil el sensor Hall se utiliza de forma frecuente, ej. en sensores de posición del cigüeñal (CKP) en el cierre del cinturón de seguridad, en sistemas de cierres de puertas, para el reconocimiento de posición del pedal o del asiento, etc.

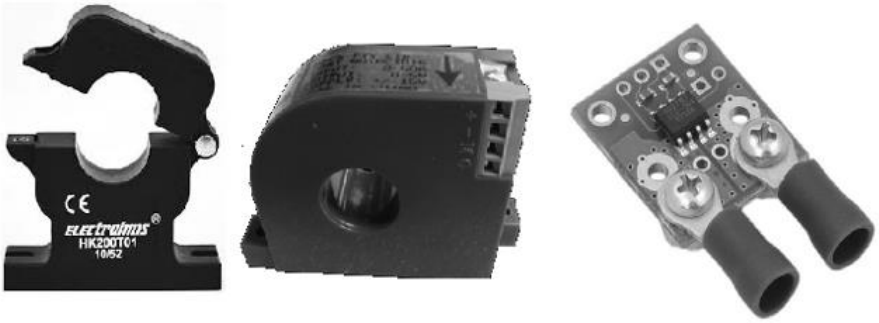

Figura 1.5: Algunos tipos de sensores de efecto Hall.
Fuente [3], [4] y [5]

1.1.7. COMPARACIÓN DE LOS SENSORES.

A continuación se muestran dos tablas comparativas el cual resume las características y los efectos para los cuatro tipos de sensores. Posteriormente una tabla cualitativa el cual resume como está el sensor en comparación con el resto de acuerdo a las distintas tecnologías utilizadas para cada uno.

Tabla 1.1: Resumen de características y efectos para cada sensor.
Fuente [1]

Sensor de Corriente.	**Características**	**Efectos**
Shunt.	De 100 a 500μΩ. Bajo costo. Inmune al problema de la saturación DC. No está aislado. Buena linealidad. No útiles para grandes corrientes.	Alta disipación de potencia en forma de calor. Necesidad de aislamiento galvánico.
Transformador de Corriente.	Capaz de medir altas corrientes. Se necesita emparejar la fase. Baja disipación comparada con la de Shunt. Proporciona aislamiento. Tiene problemas de saturación DC.	Un desfase de 0.1^0 produce un error en la facturación.
Bobina Rogowski **(Sensor di/dt).**	Tiene todas las ventajas del transformador de corriente, pero es más barato. Necesita un integrador. Inmune a la saturación DC. Buena linealidad. Bajo consumo.	Es difícil tener un integrador que sea muy estable con el tiempo.
Efecto Hall.	Son más caros. Baja linealidad. Bueno para altas corrientes.	Consumo medio. Variación alta con la temperatura.

La tabla anterior nos refleja que en caso de altas corrientes es aconsejable utilizar el sensor de efecto hall en vez del shunt, ya que este último disipa mucha potencia en forma de calor.

Los sensores más económicos son los de efecto shunt en conjunto con el Rogowski, ambos no tienen problemas con la saturación de la corriente continua.

Tabla 1.2: Comparación de las diversas tecnologías de sensores de corriente. Fuente [6]

Tecnología del sensor	**Shunt de corriente**	**Transformador de corriente**	**Sensor de Efecto Hall**	**Bobina Rogowski**
Costo	Muy bajo	Medio	Alto	Bajo
Linealidad en el rango de la medición	Muy buena	Buena	Pobre	Muy buena
Capacidad de medición de alta corriente	Muy pobre	Buena	Buena	Muy buena
Consumo de potencia	Alto	Bajo	Medio	Bajo
Problema de saturación de corriente DC	No	Sí	Sí	No
Variación de la salida con respecto a la temperatura	Medio	Bajo	Alto	Muy bajo
Problema de offset de DC	Sí	No	Sí	No
Problema de saturación e histéresis	No	Sí	Sí	No

1.2. CIRCUITOS INTEGRADOS PARA LA MEDICIÓN DE ENERGÍA ELÉCTRICA.

1.2.1. TECNOLOGÍA DE MEDICIÓN INTELIGENTE.

La figura 1.6 muestra un medidor electromecánico desarrollado por primera vez en el siglo XIX, tiene un disco giratorio y una pantalla con un contador mecánico Este tipo de medidor opera contando las revoluciones de un disco metálico que gira a una velocidad proporcional a la potencia que atraviesa el fusible principal de la caja.

Figura 1.6: Medidor electromecánico.
Fuente [7]

El primer paso en la evolución del medidor fue el reemplazo de los medidores electromecánicos con medidores electrónicos de estado sólido. Los medidores electrónicos usan componentes altamente integrados, como las familias de circuitos integrados de medición de energía ADE516x, ADE556x, ADE716x, ADE756x y ADE775x. Estos dispositivos digitalizan la corriente y voltaje instantáneo por medio de CAD[3] sigma-delta de alta resolución. Calculando el producto del voltaje y corriente se obtiene la potencia instantánea en watts. Integrando sobre el tiempo se obtiene la energía usada, la cual usualmente es medida en kilovatio-hora (kWh). Los datos de energía son mostrados en una pantalla de cristal líquido (LCD), como se muestra en la Figura 1.7.

[3] CAD: Convertidor Analógico a Digital o DAC en inglés.

Figura 1.7: Medidor de energía electrónico de estado sólido.
Fuente [7]

Los medidores electrónicos ofrecen varios beneficios, además de medir la potencia instantánea, pueden medir otros parámetros como el factor de potencia y la potencia reactiva. Los datos pueden ser medidos y almacenados a intervalos específicos, lo que permite la utilidad de ofrecer planes de precios basados en la hora del día de uso. Esto permite a los consumidores inteligentes de ahorrar dinero mediante la ejecución de los electrodomésticos, como lavadoras y secadoras, durante los períodos de menor costo, de baja demanda, y las empresas de servicios públicos pueden evitar la construcción de nuevas plantas de energía, ya que se requiere menos capacidad durante los períodos pico. Contadores electrónicos no están influenciados por imanes externos o la orientación del propio metro, por lo que son más a prueba de manipulaciones que los medidores electromecánicos. Los medidores electrónicos también son altamente confiables.

Analog Devices ha sido un actor clave en la transición de los contadores electromecánicos a electrónicos, enviando más de 225 millones de circuitos integrados

de medición de energía hasta la fecha. Según IMS Research[4], el 75% de todos los contadores de energía enviados en 2007 fueron electrónicos en lugar de electromecánicos.

Medidor electrónico abre nuevas posibilidades.

Una vez que se encuentra disponible en forma electrónica los datos del medidor, se hace factible agregar comunicaciones al medidor, lo que permite que el medidor usar la lectura automática de contadores (AMR, automatic meter reading por sus siglas en ingles) para acceder a los datos de forma remota a través del enlace de comunicación. Fabricantes de medidores han desarrollado diferentes arquitecturas de sistemas de lectura remota, ampliamente clasificados como *caminar-por*, *conducir-por*, o sistemas en red. Un sistema *conducir-por* se muestra en la Figura 1.8. En este caso, la compañía de servicios envía una camioneta con un colector de datos inalámbricos a bordo. La camioneta transitará por el barrio recogiendo los datos medidos de manera eficiente. Un sistema *conducir-por* mejora el número de medidores que un empleado de servicios públicos puede leer en un día por cinco veces, en comparación con los sistemas *caminar-por* y más de diez veces en comparación con la lectura manual del medidor. En un sistema en red, Figura 1.9, los datos medidos se alimenta a un colector de datos fijos, que normalmente se encuentra en un poste en el extremo de la calle o barrio. Los datos pasan de nuevo a la empresa de servicios públicos a través de una red troncal de banda ancha o celular.

[4] IMS Research es el proveedor líder de investigación de mercados y consultoría para la industria electrónica global.

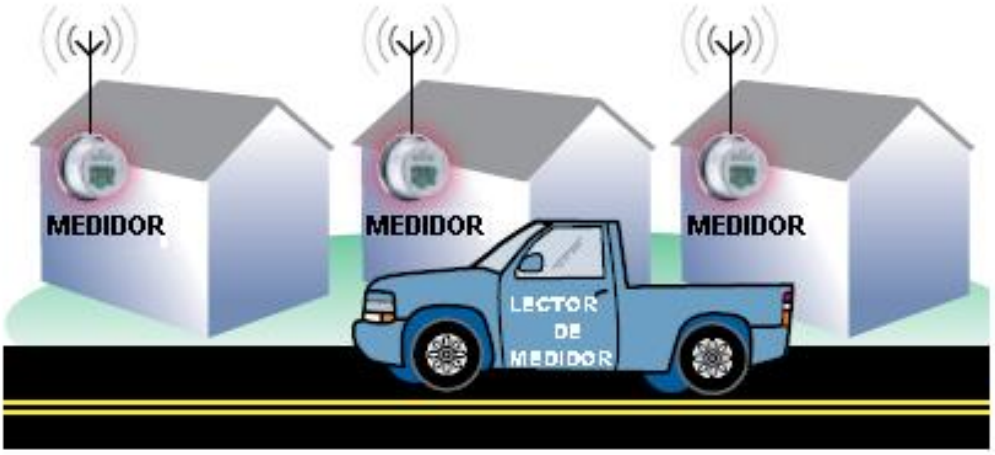

Figura 1.8: Sistema de "conducir-por".
Fuente [7]

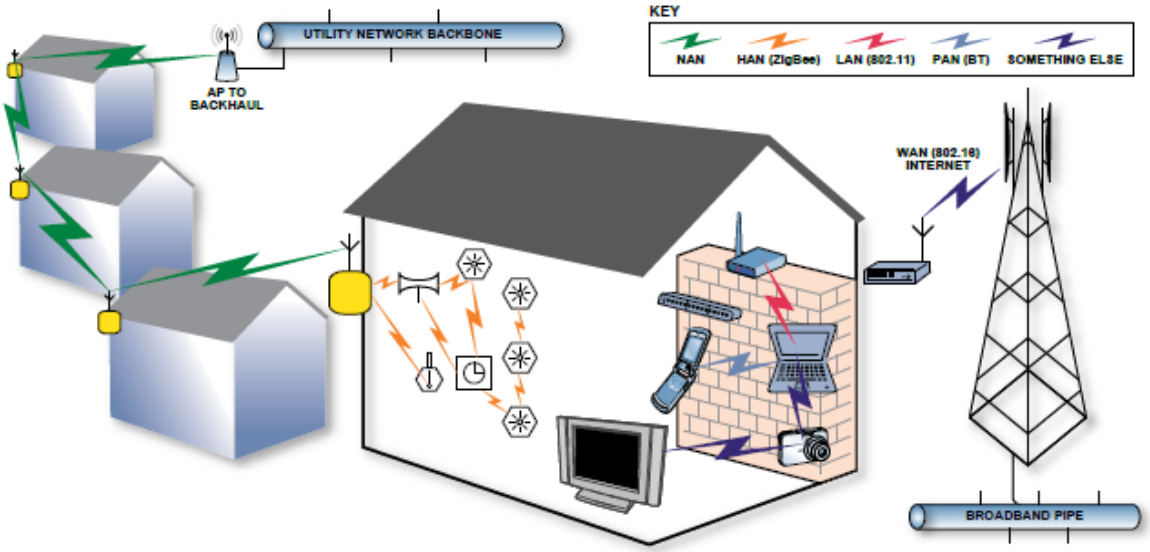

Figura 1.9: Red de área local.
Fuente [7]

1.2.2. ADE7758 (ANALOG DEVICES).

1.2.2.1. DESCRIPCIÓN GENERAL.

ADE7758 es un circuito integrado usado para le medición de energía eléctrica en sistemas trifásicos, el cual posee una interfaz serial y dos salidas de pulsos de frecuencia. Incorpora conversores A/D de segundo orden, un integrador digital, y todo el procesamiento de señal requerido para realizar cálculos de energía activa, reactiva, aparente, cálculos de tensión y corriente eficaz con una gran precisión. Es capaz de medir energía activa, reactiva y aparente en varias configuraciones trifásicas, como delta o estrella, ambas con tres o cuatro cables. Este circuito también provee un sistema de calibración por cada fase, permitiendo corregir problemas de offset, y desfases entre otros.

1.2.2.2. CANALES DE CORRIENTE ADE7758.

La figura 1.10 muestra el camino que sigue la señal de entrada en el canal de corriente. En el modo de muestreo de forma de onda, las salidas del conversor son palabras de 24 bytes cuyo formato es complemento a 2 muestreadas a un máximo de 26 kSPS. (Kilomuestras por segundo).

Con una señal analógica de entrada a plena escala de +/-0.5V, el conversor produce su máximo valor de salida cuyo valor varía entre D7AE14h y 2851Ech.

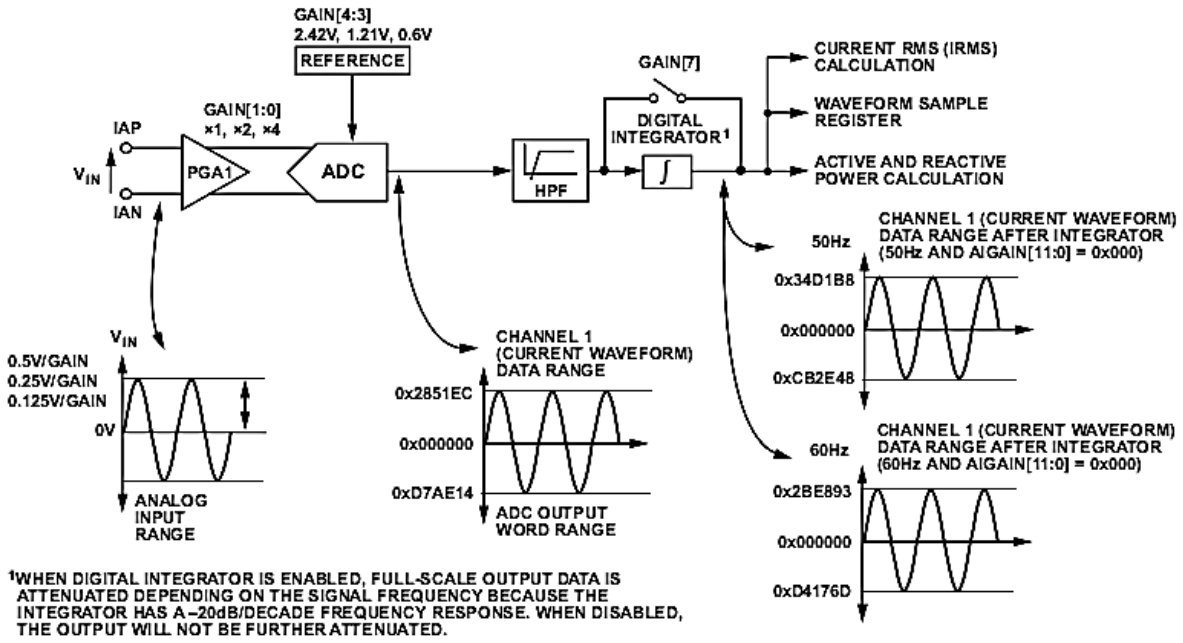

Figura 1.10: Canales de corriente del ADE7758.
Fuente [8]

1.2.2.3. CANALES DE TENSIÓN ADE7758.

La figura 1.11 muestra el procesamiento realizado para la entrada VA en el canal de tensión (igual para VB y VC).

Antes de pasar al registro de onda de forma, la salida del conversor pasa a través de un filtro pasa bajos (LPF1) con una frecuencia de corte de 260 Hz. Este filtro atenúa la señal suavemente, por ejemplo a 60 Hz esta señal es atenuada en un 3.575%. Las muestras de onda de forma son datos en complemento a 2 que varían entre 2748h y D8B8h.

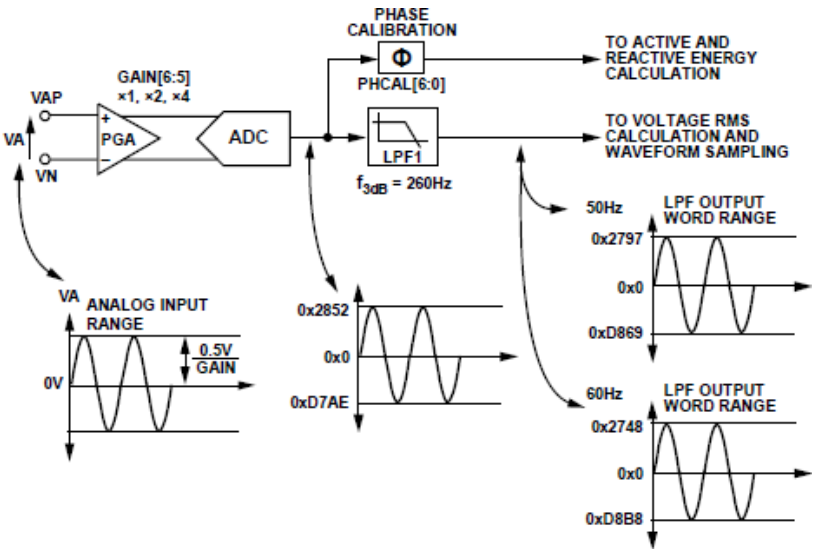

Figura 1.11: Canal de tensión del ADE7758.
Fuente [8]

1.2.2.4. PROTOCOLO DE COMUNICACIÓN ADE7758.

El *ADE7758* tiene integrada una interfaz de comunicación serial, la cual está compuesta de 4 señales: *SCLK, DIN, DOUT y* $\overline{CS}$.

***SCLK*:** Señal de reloj. Todas las operaciones de transferencia de datos son sincronizadas con esta señal de reloj.

***DOUT*:** Salida lógica que permite que los datos salgan en el flanco de subida de la señal de reloj.

***DIN*:** Entrada lógica en la cual entran los datos en el flanco de bajada de la señal SCLK.

$\overline{CS}$**:** Entrada lógica que actúa como la selección del integrado. Esta entrada es usada cuando múltiples dispositivos comparten el bus serial. Un flanco de bajada en $\overline{CS}$ coloca al *ADE7758* en modo de comunicación y debe dejarse esta entrada en activo bajo para toda la transferencia de los datos.

Todas las operaciones en el *ADE 7758* deben empezar con una escritura al registro de comunicación. El registro de comunicación es un registro de solo escritura de 8 bits. El bit más significativo determina si la siguiente transferencia de datos es lectura o escritura. Si el bit más significativo es 1 indica que la operación que se quiere llevar a cabo es de escritura mientras que si es cero la operación es de lectura. Los siete bits menos significativos contienen la dirección del registro al cual se quiere acceder.

OPERACIÓN DE ESCRITURA SERIAL.

Con el ADE7758 en modo de comunicación y la entrada *chip select* ($\overline{CS}$) en activo bajo, una escritura debe ser realizada al registro de comunicación. El bit más significativo de este byte enviado debe ser ajustado a 1, indicando que la próxima operación es de escritura de datos. El *ADE7758* empieza a leer los datos en el próximo flanco descendente del reloj serial *SCLK* y los bits restantes son leídos en el flanco descendente de los pulsos de reloj posteriores. Si ocurre otra transferencia de datos, esta debe terminar como mínimo 900nS después de que haya ocurrido la transferencia del byte anterior.

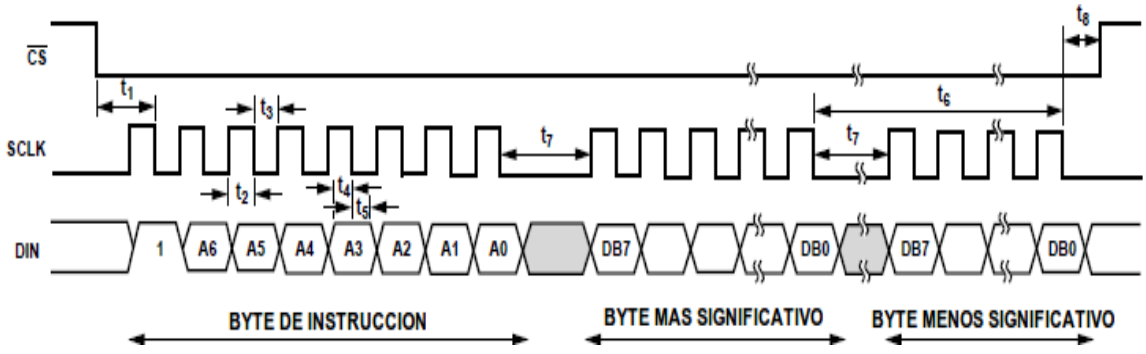

Figura 1.12: Operación de escritura serial ADE7758.
Fuente [8]

OPERACIÓN DE LECTURA SERIAL.

Durante una operación de lectura del *ADE7758* los datos son transferidos en el flanco ascendente del reloj serial. Como en el caso de la operación de escritura, una escritura debe ser realizada al registro de comunicación.

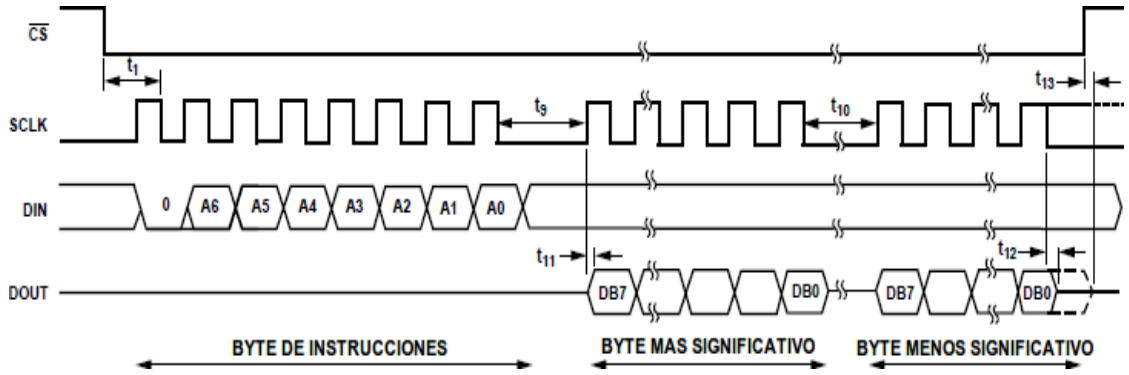

Figura 1.13: Operación de lectura serial ADE7758.
Fuente [8]

Con el *ADE7758* en modo de comunicación y $\overline{CS}$ en lógico bajo, se realiza una escritura al registro de comunicación con el bit más significativo de este ajustado a 0, indicando que se realizara una operación de lectura de datos. Los siete bits menos significativos contienen la dirección del registro que se quiere leer. En el momento en que el *ADE7758* recibe el último bit del primer byte la salida *DOUT* sale de un estado de alta impedancia y empieza a enviar datos por el bus serial. Cuando una operación de lectura se lleva a cabo, el comando de lectura no debería suceder al menos 1.1us después de la escritura en el registro de comunicación. Lo anterior se aprecia en la figura 1.13.

1.2.2.5. CÁLCULO DE POTENCIA ACTIVA ADE7758.

La potencia activa promedio sobre un número de ciclos de línea está dada por la siguiente expresión:

$$P = \frac{1}{nT}\int_0^{nT} p(t)dt = V_{rms} \times I_{rms} \times \cos\alpha$$

Donde α es la diferencia de fase entre la corriente y la tensión y *T* es el periodo del ciclo de línea.

La señal de potencia instantánea es generada multiplicando las señales de corriente y voltaje en cada fase. La componente DC de la potencia instantánea en cada fase es extraída por LPF2 (filtro pasa bajas).

La potencia activa de cada fase se acumula en el correspondiente registro de 16 bits (*AWATTHR, BWATTHR, o CWATTHR*), la señal de potencia activa obtenida a plena escala en el ADE7758 se representa en la figura 1.14.

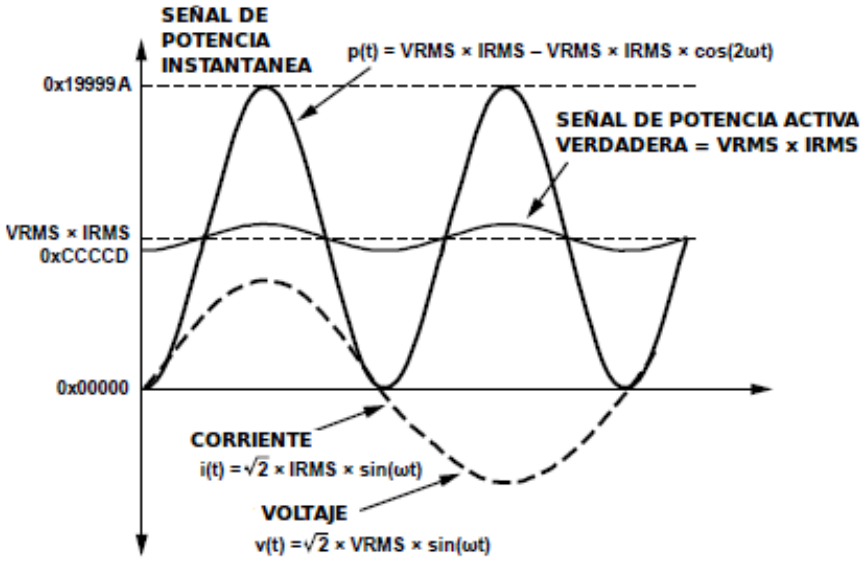

Figura 1.14: Señal de potencia activa a plena escala en el ADE7758.
Fuente [8]

El ADE7758 realiza la integración de la señal de potencia activa acumulando continuamente esta señal en registros internos de energía de 40 bits. Los registros *WATTHR (AWATTHR, BWATTHR, o CWATTHR)* representan los 16 bits más significativos de estos registros internos. La acumulación discreta en el tiempo es equivalente a la integración en tiempo continuo. La figura 1.15 muestra el esquema del proceso empleado para el cálculo de la energía activa en al ADE7758.

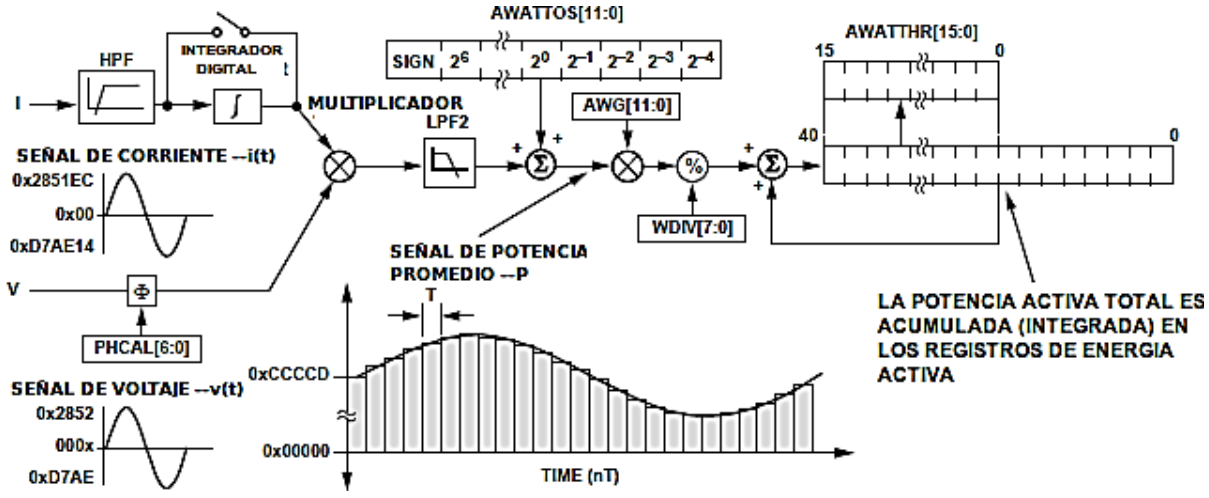

Figura 1.15: Cálculo de energía activa ADE7758.
Fuente [8]

El promedio de la señal de potencia activa es continuamente agregado al registro interno de energía. Esta adición es una operación con signo. Si la energía es negativa entonces es substraída del registro de energía activa. Los valores que se aprecian en la figura son valores cuando las entradas de tensión y corriente son a plena escala o valores nominales.

La potencia activa promedio es dividida entre el contenido de un registro divisor antes de ser agregado al registro de acumulación correspondiente *WATTHR*. Cuando el valor en el registro *WDIV [7:0]* es 0 ó 1, la potencia activa es acumulada sin división. *WDIV* es un registro sin signo, de 8 bits que es útil para aumentar el tiempo que les toma a los registros de energía en desbordarse.

Con señales de entrada a plena escala el tiempo mínimo que le toma a estos registros en desbordarse depende del registro de ganancia *xWATT GAIN*. Cuando este registro presenta valores de *7FFh,000h y 800h* los tiempos mínimos son 0.13, 0.52 y 0.79 segundos respectivamente.

Se puede activar una interrupción en el *ADE7758* para que avise cuando un registro de acumulación de energía está lleno hasta la mitad, con el fin de leerlo antes que este registro se desborde y se pierdan los datos. Otra forma de leer estos registros es activar el bit RSTREAD con el fin de que estos registros vuelvan a cero después de una lectura.

El periodo de muestreo en tiempo discreto para la acumulación de energía es 0.4us (4/*CLKIN*), por lo tanto se toman 2500 kSPS (kilo muestras por segundo). Si se supone una carga estable, el tiempo mínimo que transcurre antes que los registros de energía se desborden está dado por la siguiente ecuación:

$$Tiempo_{minimo} = \frac{0xFF,FFFF,FFFF}{0xCCCCD} \times 0.4\,\mu S = 0.524\ segundos$$

Donde 0x*FF,FFFF,FFFF* es el valor máximo que puede almacenar el registro interno de energía y 0x*CCCCD* es la máxima salida del filtro pasa bajas.

Para el cálculo de la potencia activa se divide el contenido del registro de energía entre el tiempo de acumulación de este registro, dicho tiempo se calcula de la siguiente forma:

$$Tiempo_{acumulacion} = Tiempo_{minimo} \times WDIV[7:0]$$

Donde es el tiempo de acumulación cuando el registro de ganancia es y que corresponde a 0.52 segundos, y *WDIV[7:0]* es el factor de escalamiento el cual puede ser de 255 veces máximo.

El *ADE7758* también provee un pin de salida (*APCF*) capaz de entregar información de la energía activa a través de pulsos de frecuencia, los cuales son proporcionales a la energía activa medida.

1.2.2.6. CÁLCULO DE POTENCIA REACTIVA ADE7758.

La potencia reactiva promedio sobre un número de ciclos de línea está dada por la siguiente expresión:

$$Q = \frac{1}{nT}\int_0^{nT} q(t)\,dt = V_{rms} \times I_{rms} \times \sin\alpha$$

Al igual que para la potencia activa, la componente DC de la señal de potencia reactiva instantánea en cada fase es extraída por un filtro pasa bajas. Este proceso es ilustrado en la figura 1.16. La potencia reactiva de cada fase es acumulada en el registro correspondiente de 16 bits *(AVARHR, BVARHR, CVARHR).*

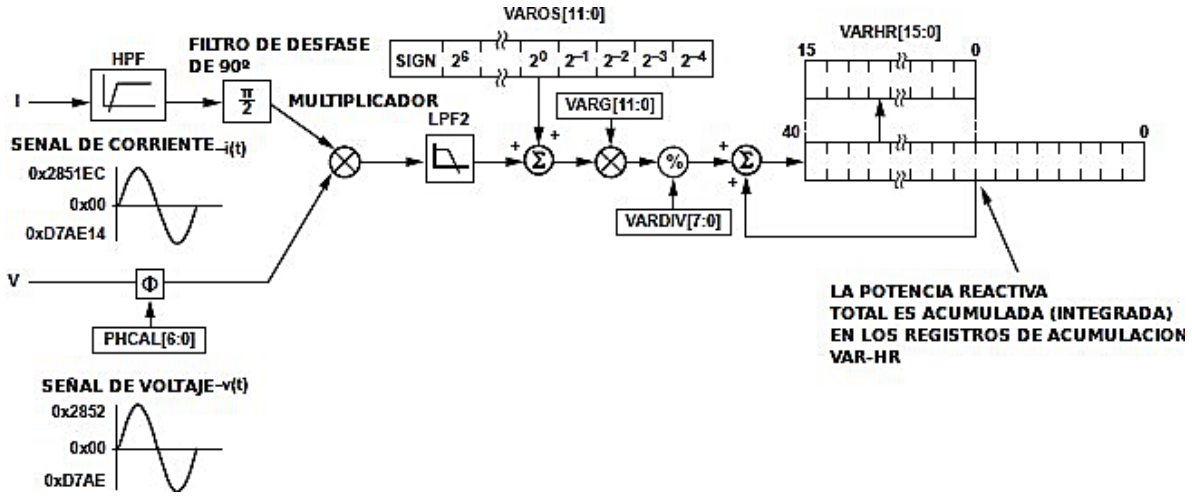

Figura 1.16: Cálculo de energía reactiva ADE7758.
Fuente [8]

La señal de potencia reactiva es acumulada constantemente en los registros internos de energía. Esta adición es una operación con signo por lo tanto energías negativas son substraídas. La potencia activa promedio es dividida entre el contenido de un registro divisor antes de que esta sea agregada al registro de energía con el fin de aumentar el tiempo en el cual se desborda el registro de energía.

El pin 17 (*VARCF*) de *ADE7758* es una salida que emite pulsos proporcionales a la energía reactiva total.

1.2.2.7. CÁLCULO DE POTENCIA APARENTE ADE7758.

ADE7758 usa el método de aproximación aritmética para calcular la potencia aparente.

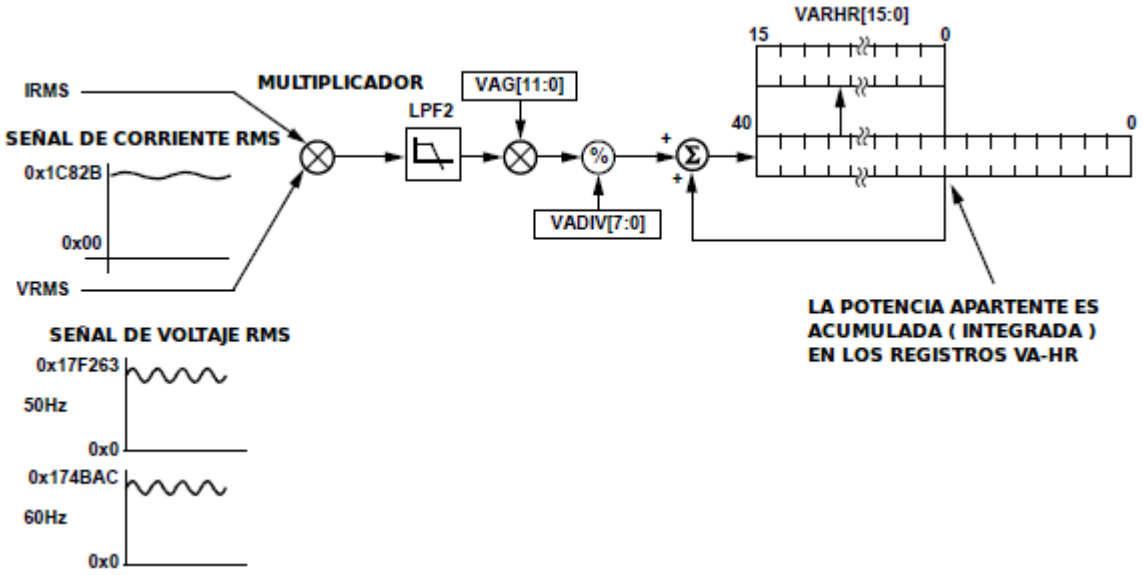

Figura 1.17: Cálculo de potencia aparente ADE7758.
Fuente [8]

Los valores eficaces de la tensión y la corriente son multiplicados en cada fase para producir la potencia aparente de la correspondiente fase. La salida del

multiplicador pasa por un filtro pasa bajas para obtener la potencia aparente promedio.

A diferencia de la potencia activa y reactiva, no existen registros para compensar el *offset* existente en canal de potencia aparente. Esto se debe a que la compensación de *offset* que se le hace a los registros de tensión y corriente es suficiente.

El proceso para calcular la energía aparente es el mismo explicado anteriormente para la energía activa y reactiva. El único aspecto diferente es el tiempo mínimo de acumulación de energía, el cual es mayor en este caso y se muestra a continuación:

$$Tiempo_{mínimo} = \frac{0x1FFFFFFFFF}{0xB9954} * 0.4\mu s = 1.157\ segundos$$

Donde 0xB9954 es la máxima salida del filtro.

Utilizando el registro *VADIV* se puede aumentar el tiempo de acumulación de la energía aparente en los registros respectivos.

Ajustando el bit 7 en el registro *WAVMODE* se asegura que la salida de pulsos en el pin 17 muestra información acerca de la potencia aparente y no de la potencia reactiva.

1.2.2.8. CÁLCULO DE CORRIENTE ADE7758.

La figura 1.18 muestra el proceso para calcular el valor la tensión *rms* por parte del *ADE7758* para una fase. El mismo procedimiento es usado para las dos fases restantes.

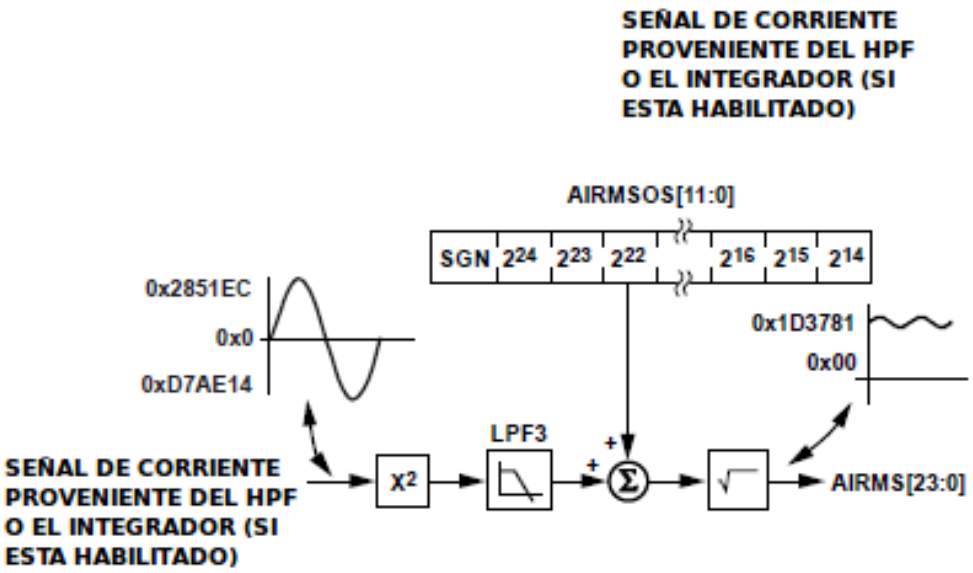

Figura 1.18: Cálculo de corriente ADE7758.
Fuente [8]

El filtro LPF3 extrae el promedio de la señal de corriente al cuadrado, la señal resultante se suma con un registro que corrige el *offset* (*AIRMSOS*), para después extraer la raíz cuadrada y guardar el resultado en el registro de 24 bits *AIRMS*.

Con las señales de entrada a plena escala, el conversor produce un código de salida aproximado de 1D3781h. El ADE7758 tiene registros capaces de remover el *offset* presente en cualquiera de las fases (*AIRMSOS, BIRMSOS, CIRMSOS*). Un *offset* puede existir debido a ruidos de entrada que son integrados en la componente DC cuando se eleva al cuadrado la corriente.

1.2.2.9. CÁLCULO DE TENSIÓN ADE7758.

Con las señales análogas de entrada a plena escala (0.5V), el filtro LPF1 produce un código de salida aproximado de +/-9,372d a 60 Hz. Posteriormente esta salida es elevada al cuadrado y pasa nuevamente por un filtro pasa bajas LPF3 con el fin de extraer el valor promedio de la señal de tensión. Luego es extraída la raíz cuadrada y

finalmente se suma la señal resultante con un registro que corrige el offset que se presente en el canal de tensión de cada fase. El registro xVRMSGAIN[5] es usado para escalar las salidas de los conversores A/D en +/- 50%.

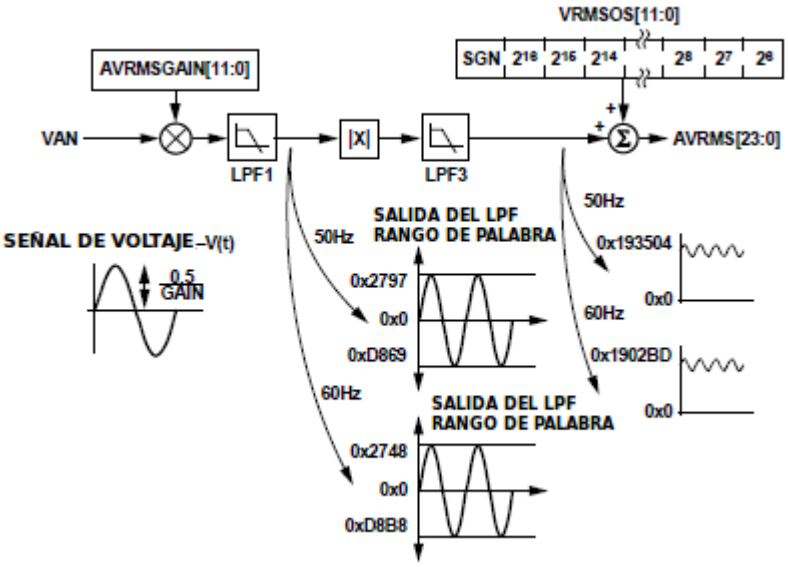

Figura 1.19: Cálculo de tensión ADE7758.
Fuente [8]

El error típico en la medición de la tensión RMS es de 0.5% y esta medición tiene una ancho de banda de 260Hz.

1.2.2.10. INTERRUPCIONES.

Las interrupciones en el *ADE7758* son manejadas a través del registro de estado de interrupción, (*STATUS [23:0],* dirección 19h) y del registro mascara de interrupción (*MASK [23:0],* dirección 18h). Cuando un evento de interrupción ocurre

[5] "x" hace referencia a la fase A, B o C.

en el *ADE7758*, la bandera correspondiente en el registro estado de interrupción cambia a 1 lógico. Si el bit de mascara para esta interrupción es un 1 lógico, entonces la salida lógica pasa a activa bajo. Para determinar la fuente de la interrupción, el microcontrolador debe realizar una lectura del registro *RSTATUS*. Después de realizada la lectura del registro la salida vuelve a su estado normal activo alto.

1.3. CAN (CONTROLLER AREA NETWORK).

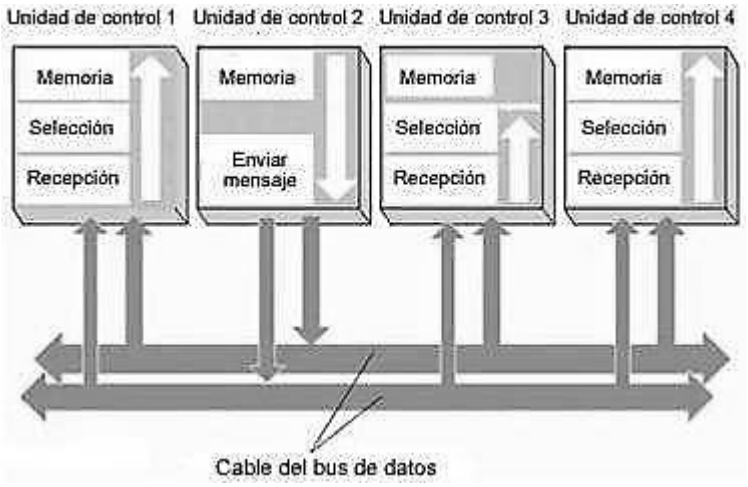

Figura 1.20: Ejemplo de envío de paquete en CAN.
Fuente [9]

El bus CAN (Controlador de Área de Red) es un bus de datos de comunicación serie, empleado para su aplicación en sistemas **distribuidos** en tiempo real. Originalmente el bus CAN fue desarrollado para aplicaciones en la industria automotriz, pero debido a sus características, robustez y excelente relación calidad/precio, CAN fue adoptado para aplicaciones industriales y de control.

El protocolo de comunicación CAN fue especificado originalmente por la compañía alemana Robert Bosch para aplicaciones críticas en tiempo real.

El protocolo CAN es un protocolo CSMA/CD. El CSMA6 significa que cada **Controlador Inteligente** en la red debe monitorear el bus esperando un tiempo de inactividad antes de tratar de enviar un mensaje (Detección de Portadora), una vez ocurre cada controlador tiene igual oportunidad de transmitir un mensaje (Múltiple Acceso). El CD7 significa que si dos controladores en la red comienzan a transmitir al mismo tiempo, detectaran una "colisión" y tomaran la acción apropiada para evitarla y mantener la integridad de los datos. En el protocolo CAN, se utiliza un método de arbitraje no destructivo a nivel de bit, lo que significa que los mensajes permanecen intactos después que el arbitraje es completado incluso si se detectan colisiones. El arbitraje toma lugar sin corromper o retardar el mensaje de mayor prioridad. Este protocolo es utilizado para transportar las tramas del protocolo MODBUS a través de los controladores Inteligentes.

Los motivos principales que han llevado a la elección de este protocolo son las siguientes:

- Tratamiento de errores muy eficaz. Es un protocolo muy robusto frente a los problemas de ruido, ya que fue diseñado para entornos industriales.
- Es un sistema que se ha ido adaptando a otros campos, lo que ha hecho que su uso se haya extendido. Por ello, su coste es bajo frente al coste de otros posibles sistemas.
- Posibilidad de implementación con dispositivos relativamente simples y baratos como por ejemplo los PIC's que usamos.
- Facilidad de desarrollo y gran cantidad de fabricantes en el mercado de dispositivos CAN.
- Su sistema de prioridades "no destructivo", que permite que en caso de que se transmitan dos mensajes simultáneamente, el de mayor prioridad no se destruya y llegue a su destino sin ningún tipo de retardo añadido.

[6] Carrier Sense Multiple Access : múltiple acceso con detección de portadora
[7] Collision Detect : Detección de Colisión

- Su sistema de identificación de mensajes, que consiste en que los nodos no tienen realmente una dirección, sino que se programan con un sistema de filtros para que acepten un determinado tipo de mensajes (Ver la figura 1.20), es decir, **es un sistema basado en tipos de mensajes, no en direcciones**, lo que hace que se puedan añadir nuevos nodos sin tener que reconfigurar el resto de los nodos.

- Su comportamiento en tiempo real: El método empleado por el protocolo CAN asocia cada mensaje a ser enviado con una prioridad determinada, y usa un mecanismo especial de arbitraje para asegurar que el mensaje de mayor prioridad sea el mensaje transmitido. La prioridad de un mensaje es un número único y puede ser usado como el identificador del mensaje. Es por eso que a la prioridad se le denomina también identificador del mensaje.

En cuanto a inconvenientes:

- Protocolo complejo.
- Velocidad limitada por la longitud de la red (Ver la figura 1.21). Según el estándar, la velocidad máxima que puede alcanzar el bus CAN es de 1Mbps y esta se puede alcanzar hasta con una longitud de red de 40 metros.

Longitud del bus	Velocidad en bits/s	Tiempo máximo de transmisión*
Hasta 25 m	1 Mbit/s	129 µs
Hasta 100 m	500 Kbit/s	258 µs
Hasta 500	125 Kbit/s	1032 ms
Hasta 1000 m	50 Kbit/s	2580 ms

* mensajes de 129 bits de longitud

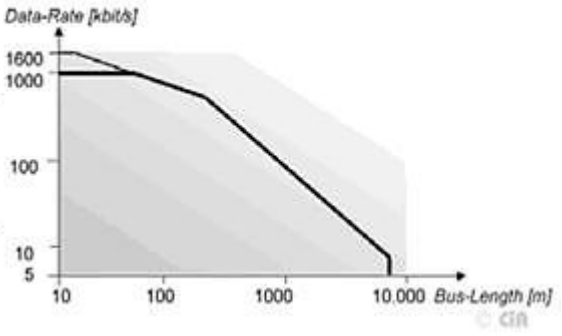

Figura 1.21: Velocidad del bus CAN en función de la longitud.
Fuente [9]

Para el desarrollo de la red CAN se usa en el estándar ISO 11898 (Organización Internacional para la Estandarización), en aspectos concretos del protocolo CAN como son los niveles del bus, implementación de nodos, etc.

Dado que el protocolo CAN sólo define dos niveles del Modelo OSI (Ver la figura 1.22), es necesario que se defina también un nivel alto de aplicación.

Figura 1.22: Modelo OSI.
Fuente [9]

1.3.1. CAPA FÍSICA.

Se definen los parámetros a nivel físico (niveles de señal, corriente, sincronización, etc.). CAN no tiene declarada una especificación como tal, pero los estándares ISO 11898 establecen las características que deben de cumplir las aplicaciones para la transferencia en alta y baja velocidad.

CAN especifica dos estados lógicos: Recesivo y Dominante. La ISO-11898 define un voltaje diferencial para representar los estados recesivo y dominante. En el estado Recesivo (lógico '1' en la entrada TXD del MCP2551). El voltaje diferencial en CANH y CANL es menor que el umbral mínimo (<0.5V entrada receptora o <1.5V salida transmisora). En el estado dominante (lógico '0' en la entrada TXD del MCP2551), el voltaje diferencial en CANH y CANL es más grande que el umbral mínimo. ISO-11898-2 no especifica la conexión eléctrica. Sin embargo, la especificación requiere resistores terminales de 120Ω en cada final del bus.

La Terminación del Bus con resistores minimiza la reflexión de la señal en el bus lo que permite lograr las longitudes máximas previamente vistas en la figura 1.21. Existen diferentes métodos de terminación del bus empleados para incrementar la longitud, disminuir la reflexión de la señal y aumentar el desempeño EMC[8], ver Figura 1.23, en la tabla 1.3 se da una breve descripción de los métodos de terminación.

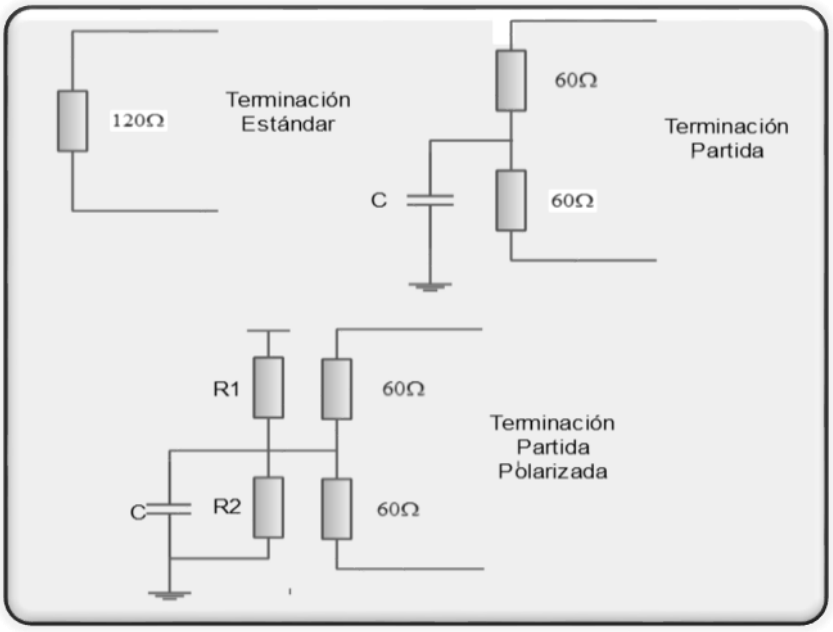

Figura 1.23: Métodos de terminación del bus CAN.
Fuente [10]

[8] **EMC**: Electromagnetic Compatibility. Es una rama de la Ingeniería eléctrica que estudia la generación, propagación y recepción inintencionada de energía electromagnética, así como los efectos no deseados de esa energía. El objetivo de la EMC es el funcionamiento correcto en el mismo entorno de diferentes equipos que producen fenómenos electromagnéticos cuando operan.

Tabla 1.3: Tipos de terminaciones del bus CAN.
Fuente [10]

TIPO DE TERMINACION	DESCRIPCION
Terminación estándar	Utiliza resistores de 120Ω en las terminaciones del bus.
Terminación partida	El resistor de 120Ω en cada extremo del bus es partida en 2 resistores de 60Ω, con un capacitor de bypass entre los dos resistores y tierra. Los dos resistores deben tener el mismo valor tanto como sea posible.
Terminación partida polarizada	Método utilizado para mantener el voltaje de modo común a un valor constante, incrementando así el rendimiento EMC. Este circuito es el mismo que la terminación partida con un divisor de voltaje adicional para alcanzar un voltaje de VDD/2 entre los dos resistores de 60Ω. Ver figura 1.32.

La información circula por dos cables trenzados que unen todas las unidades de control que forman el sistema. Esta información se trasmite por diferencia de tensión entre los dos cables, de forma que un valor alto de tensión representa un 1 y un valor bajo de tensión representa un 0. La combinación adecuada de unos y ceros conforman el mensaje a trasmitir.

En un cable los valores de tensión oscilan entre 0 V y 2.25 V, por lo que se denomina cable L (Low), y el otro cable, llamado cable H (High) tiene niveles de tensión entre 2.75 V y 5 V (Ver la figura 1.24).

En caso de que se interrumpa la línea H o que se derive a masa, el sistema trabajará con la señal de Low con respecto a masa. En el caso de que se interrumpa la línea L, ocurrirá lo contrario. Esta situación permite que el sistema siga trabajando con uno de los cables cortados o comunicados a masa, quedando fuera de servicio solamente cuando ambos cables se cortan.

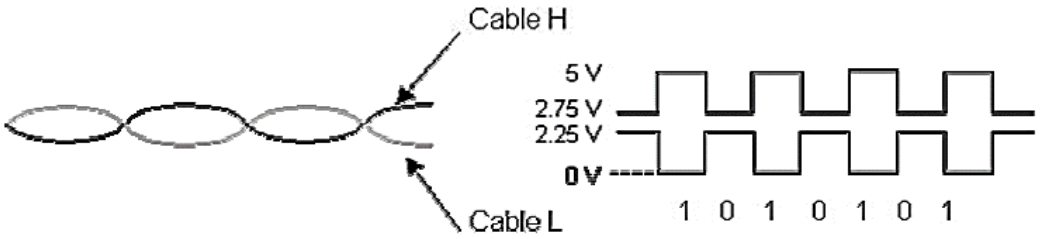

Figura 1.24: Modelo Niveles de tensión utilizados en CAN.
Fuente [11]

El valor dominante en el bus CAN se produce cuando tenemos el bit ′0′ mientras que el valor recesivo se produce con el bit a ′1′.

Es importante tener en cuenta que el trenzado entre ambas líneas sirve para anular los campos magnéticos, por lo que no se debe modificar en ningún caso ni el paso ni la longitud de dichos cables.

1.3.2. CAPA DE ENLACE DE DATOS (LLC Y MAC).

La MAC (Media Access Control) es la responsable de:

- **Tipo de trama que se envía**: Para la transición y control de mensajes CAN se definen cuatro tipos de tramas:
    1. De datos.
    2. Remota (transmitido por una unidad de bus que requiere la transmisión de una trama de datos con el mismo identificador).
    3. De error (transmitido por cualquier unidad que detecta un error en el bus).
    4. De sobrecarga (usado para dar un retraso extra entre las tramas de datos y remotas).

Tanto las tramas de datos y las tramas remotas tienen dos formatos. Un formato es el llamado Estándar y el otro es el Extendido, y se diferencian en el número de bits que tiene el identificador (la estándar tiene 11 bits y la extendida tiene 29 bits) (Ver la figura 1.25). Las tramas de datos y las remotas se separan de tramas precedentes mediante espacios entre tramas (interframe space).

Dentro de las tramas de datos y remotas podemos encontrarnos con diferentes campos (Ver la figura 1.26):

- Inicio de trama: consiste en un bit a '0'.
- Campo de arbitración: campo donde se introduce el identificador.
- Campo de control: campo donde se indica la longitud de los datos.
- Campo de datos: campo donde se introducen los datos (como máximo 8 bytes).
- Campo CRC: campo de comprobación de errores.
- Campo ACK: Consta de dos bits en 'recesivos'.
- Fin de trama: consiste en 7 bits recesivos.

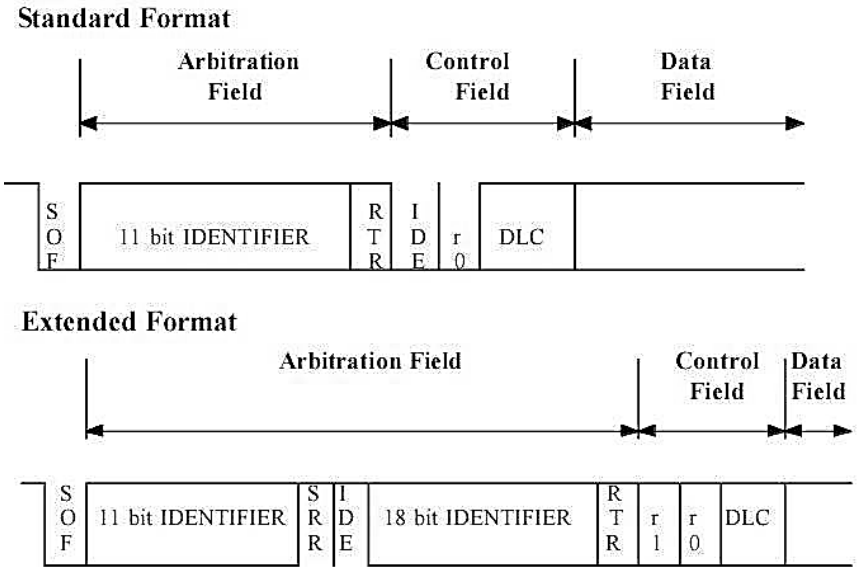

Figura 1.25: Formato estándar y extendido de las tramas CAN. Fuente [9]

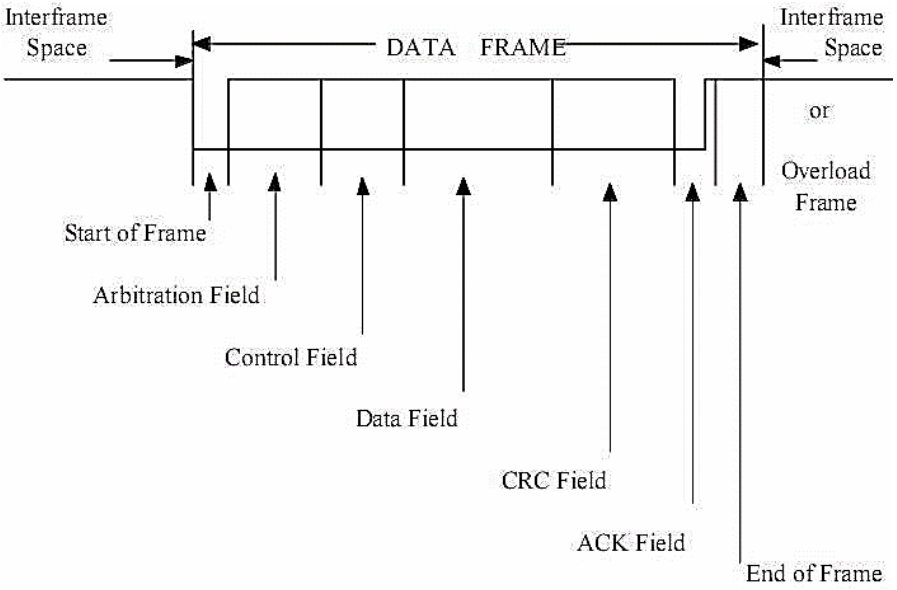

Figura 1.26: Campos de las tramas CAN.
Fuente [9]

La única diferencia entre la trama de datos y la trama remota es que esta última no contiene el campo de datos.

- **Arbitración:** Cuando el bus esta idle (libre), cualquier unidad puede transmitir un mensaje. Si dos o más dispositivos quieren transmitir mensajes al mismo tiempo, el conflicto de quien accederá al bus será resuelto por el "árbitro" utilizando el identificador. Este mecanismo garantiza que no se pierde información ni tampoco tiempo. En el caso de una trama de datos y una remota con el mismo identificador, la trama de datos prevalece sobre la trama remota.

  Durante la arbitración cada transmisor compara el nivel del bit transmitido con el que monitoriza el bus: Si esos niveles son iguales continua enviando, si son diferentes y lo que hemos enviado es un bit recesivo '1' y monitorizamos un bit dominante '0', la unidad habrá perdido la arbitración y deberá dejar de transmitir (Ver la figura 1.27).

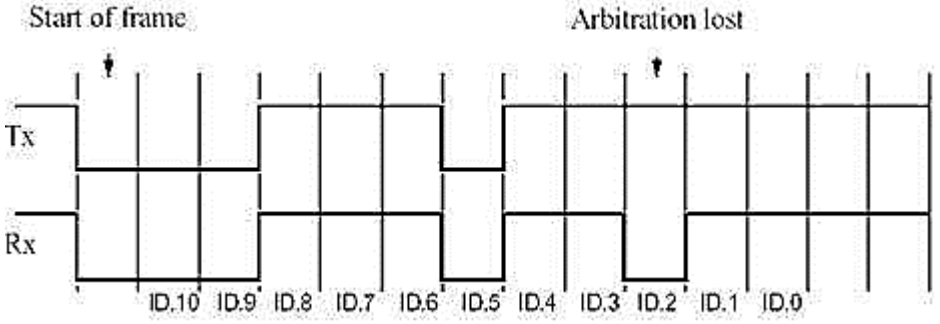

Figura 1.27: Ejemplo de arbitración en CAN.
Fuente [9]

- **ACK:** Un receptor que recibe un mensaje correctamente se lo notifica al transmisor poniendo el bit del campo ACK a ’0’ (dominante), de forma que el transmisor que esta´ todavía trasmitiendo reconoce que al menos alguien ha recibido un mensaje escrito correctamente. De no ser así, el transmisor interpreta que su mensaje presenta un error.

- **Detectar errores:** En cuanto a la detección y manejo de errores, un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red. Todo error detectado por un nodo se notifica inmediatamente al resto de los nodos.

- **Definir el método de acceso:** El método de acceso al medio utilizado es el de Acceso Múltiple por Detección de Portadora, con Detección de Colisiones y Arbitra je por Prioridad de Mensaje (CSMA/CD+AMP, Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority).
  De acuerdo con este método, los nodos en la red que necesitan transmitir información deben esperar a que el bus esté libre (detección de portadora). Cuando se cumple esta condición, dichos nodos transmiten un bit de inicio (acceso múltiple). Cada nodo lee el bus bit a bit durante la transmisión de la trama y comparan el valor transmitido con el valor recibido; mientras los

valores sean idénticos, el nodo continúa con la transmisión; si se detecta una diferencia en los valores de los bits, se lleva a cabo el mecanismo de arbitraje.

La LLC (Logical Link Control) es la capa que está relacionada con:

- Filtrado de mensajes: Como se ha comentado anteriormente, los dispositivos no tienen realmente una dirección, sino que se programan con sistema de filtros para acepten un determinado mensaje.

- Proceso de solución de errores (reenvío): El protocolo CAN tiene cinco métodos de repaso de errores, tres en el nivel de mensaje y dos en el nivel del bit. Si un mensaje tiene alguno de estos errores, no se aceptará y se generará una trama de error para que el resto de los nodos no hagan caso del mensaje defectuoso, y para que el nodo que transmite vuelva a enviar el mensaje.

1.3.3. CAPA DE SUPERVISOR.

Un sistema de bus serie presenta el problema de que un nodo defectuoso puede bloquear el funcionamiento del sistema completo. Cada nodo activo transmite una bandera de error cuando detecta algún tipo de error y puede ocasionar que un nodo defectuoso pueda acaparar el medio físico.

Para eliminar este riesgo el protocolo CAN define un mecanismo autónomo para detectar y desconectar un nodo defectuoso del bus. Dicho mecanismo se conoce como aislamiento de fallos o "Fault Confinement".

1.3.4. CAPA DE APLICACIÓN.

Existen diferentes estándares que definen la capa de aplicación, algunos son muy específicos y están relacionados con sus campos de aplicación.

Entre las capas de aplicación más utilizadas cabe mencionar CAL, CANopen, DeviceNet, SDS (Smart Distributed System), OSEK y CANKingdom.

1.4. RS232 (RECOMMENDED STANDARD 232).

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera a los equipos de varios fabricantes comunicarse entre si. La EIA (Electronics Industry Association) elaboro la norma RS-232, la cual define la interfaz mecánica, los pines, las señales y los protocolos que debe cumplir la comunicación serial.

El RS232 es un protocolo de comunicación serie orientado a caracteres, es decir, un protocolo donde toda la información es enviada por un solo canal bit a bit (un canal para enviar información y otro para recibirla), y donde lo que se envían son caracteres. Por ejemplo, si queremos enviar el número 123, primero tendremos que enviar el carácter 1, seguidamente el 2 y para finalizar el 3, y no el byte que represente el número 123.

Este protocolo está diseñado para distancias cortas, de unos 15 metros más o menos, y se puede trabajar de forma asíncrona o síncrona y con tipos de canal simplex, half-duplex y full-duplex[9].

[9] Un canal simplex consiste en una comunicación unidireccional, deshabilitando la respuesta del receptor. Un canal half-duplex permite la comunicación en ambos sentidos pero no simultáneamente, mientras que un canal full-duplex permite una comunicación bidireccional simultáneamente.

Una conexión RS232 está definida por un cable desde un dispositivo al otro. Hay 25 conexiones en la especificación completa pero en la mayoría de los casos se utilizan menos de la mitad. Los conectores más utilizados son los DB9 y los DB25 (Ver la figura 1.28).

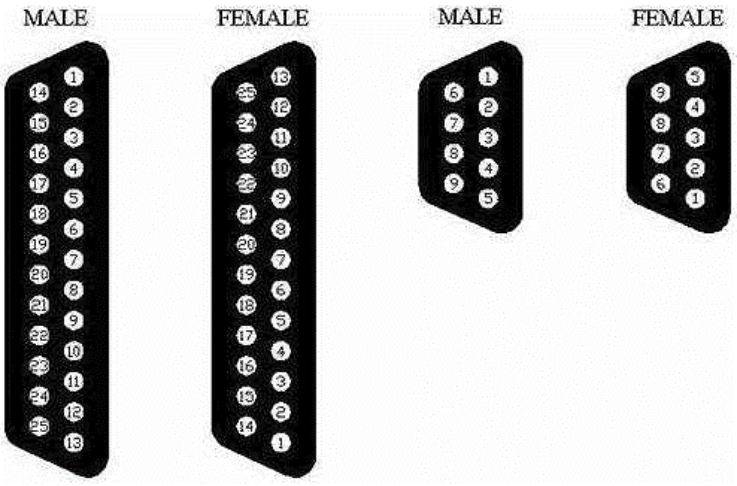

Figura 1.28: Conectores DB-25 y DB-9 en RS232.
Fuente [9]

En la tabla 1.4 se puede observar las señales más comunes en RS232 según los pines del conector asignados:

- GND: Valor a 0V.
- TD: Línea de datos del transmisor al receptor.
- RD: Línea de datos del receptor al transmisor.
- DTR: Línea por donde el receptor informa al transmisor que está vivo y bien.
- DSR: Línea por donde el transmisor informa al receptor que está vivo y bien.
- RTS: Línea en la que el transmisor indica que quiere enviar algo al receptor.
- CTS: Línea en la que se informa que el receptor está preparado para recibir datos.

- DCD: Línea por la que el receptor informa al transmisor que tiene una portadora entrante.
- RI: Línea en la que se indica que se ha detectado una portadora.

Tabla 1.4: Señales más comunes en RS232 y sus respectivos pines en los conectores.
Fuente [9]

SEÑAL	DB-25	DB-9
GND	7	5
Transmisión de datos (TD)	2	3
Recepción de datos (RD)	3	2
Terminal de datos preparado (DTR)	20	4
Datos preparados (DSR)	6	6
Petición de envío (RTS)	4	7
Limpieza de envío (CTS)	5	8
Portadora de datos detectada (DCD)	8	1
Indicador de tono (RI)	22	9

La conexión más sencilla se puede realizar con 3 cables (TD, RD, y GND). En este trabajo utiliza esta configuración de 3 cables.

Los parámetros a configurar en una comunicación RS232 son los siguientes:

- Protocolo serie (**número de bits - paridad - bits de parada**): La paridad se utiliza para poder comprobar la calidad de los datos recibidos. Los bits de datos pueden estar entre los 5 bits y los 8, y los bits de parada consisten en uno o dos bits puestos a ’1’. En la figura 1.29 se muestra una configuración 7N1.

- **Velocidad del puerto:** RS232 puede transmitir los datos a unas velocidades determinadas (normalmente entre 4800 y 115200 bps).

- Protocolo de **control de flujo**: El control de flujo puede ser mediante hardware gracias al llamado 'handshaking' entre las Líneas RTS y CTS, o por software mediante el XON/XOFF[10]. En nuestro proyecto no utilizaremos control de flujo.

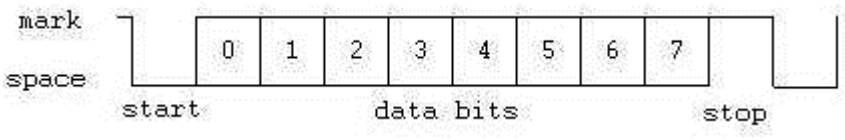

Figura 1.29: Configuración 7N1 del RS232.
Fuente [9]

1.5. SPI (SERIAL PERIPHERAL INTERFACE).

El SPI es un bus full-duplex, síncrono y serial desarrollado por Motorola. También puede encontrarse con el nombre de Microwire, propiedad de National SemiConductors.

Al ser un bus serial su número de hilos es reducido en comparación a los buses paralelos. Si bien estos últimos son más rápidos, la eficiencia y simplicidad que un bus serial puede ofrecer hace que esta diferencia de velocidad sea menos importante.

1.5.1. DESCRIPCIÓN DEL BUS.

SPI es un bus que establece la comunicación entre master y esclavo mediante 4 tipos de hilos que contiene señales diferentes:

[10] En el XON/XOFF, cuando el receptor quiere que el transmisor pare su envío de datos envía XOFF, mientras que cuando el receptor quiere que envíe más datos, envía XON.

- **MISO**, Master In/Slave Out. Esta línea es una de las dos líneas unidireccionales.
  A través de esta línea se realiza la transmisión de datos de forma unidireccional desde la salida del Esclavo a la entrada del Maestro. Cuando el dispositivo no ha sido seleccionado para la comunicación, esta línea es puesta en un estado de alta impedancia para evitar interferencias.

- **MOSI**, Master Out/Slave In. Esta línea es la segunda de las dos líneas unidireccionales. A través de esta línea se realiza la transmisión de datos de forma unidireccional desde la salida del master a la entrada del esclavo. El dispositivo master pone los datos sobre la línea MOSI medio ciclo antes del final del flanco de alta impedancia para evitar interferencias.

- **SCLK**, Es el reloj del bus con el que los dispositivos sincronizarán el flujo de datos a través de las líneas.

Se pueden configurar 2 parámetros que definen 4 modos de sincronización. Estos parámetros son:

- CPOL (Clock polarity): Determina si el estado IDLE de la línea SCLK es en nivel bajo (CPOL = 0) o en nivel alto (CPOL = 1). No tiene efecto significativo en el formato de transferencia.

- CPHA (Clock Phase): Determina en cuál flanco del reloj los datos son leídos o escritos. Si CPHA = 0 los datos sobre la línea MOSI son puestos en el primer flanco de reloj y los datos sobre la línea MISO son leídos en el segundo flanco de reloj. Cuando CPHA = 1 sucede lo contrario, la transferencia de datos sucede en el segundo flanco de reloj.

Por lo tanto, según cómo se combinen estos dos parámetros, tendremos 4 modos diferentes de trabajo.

Es muy importante que todos los dispositivos dentro del bus trabajen con el mismo modo.

- $\overline{CS}$ **o** $\overline{SS}$, Chip select o Slave Select. Los dispositivos en SPI no se seleccionan por software utilizando direcciones, como ocurre en I2C. En este protocolo es necesario utilizar una línea más llamada Chip Select, ($\overline{CS}$) o Select Slave ($\overline{SS}$).

  De forma que si estamos utilizando 3 dispositivos, el bus estará compuesto por 2 hilos de transmisión de datos, 1 de reloj y 3 de $\overline{CS}$ cómo se puede observar en la figura 1.30.

Si sólo se utiliza un Master y un Esclavo como en el caso de nuestro nodo de comunicaciones, las líneas CS no serán necesarias.

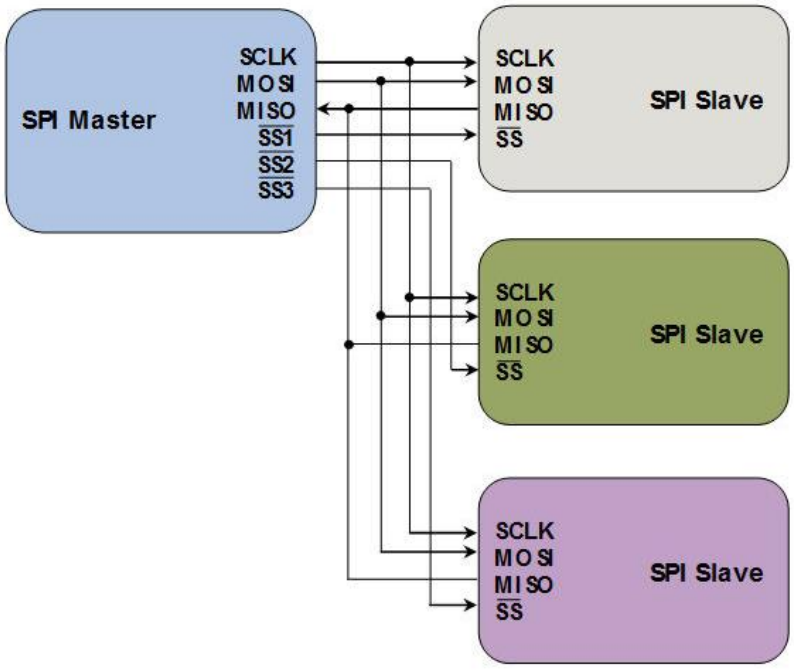

Figura 1.30: Conexión del CS de SPI a los diferentes dispositivos.
Fuente [9]

1.5.2. FUNCIONAMIENTO DEL BUS.

Cuando se la establece comunicación, el bus sólo puede ser ocupado por un Master y un esclavo. Cualquier dispositivo que no haya sido seleccionado deberá deshabilitarse por medio del chip select para evitar interferencia.

La selección de los dispositivos es muy sencilla: El master pone a nivel lógico bajo '0' el chip select del dispositivo en cuestión. El resto pasan a modo de alta impedancia para no interferir la comunicación. A partir de este momento se inicia la comunicación entre los dispositivos. Es algo parecido a la condición de inicio de I2C.

Una vez seleccionados los dispositivos, los datos pueden ser transferidos hasta una velocidad máxima de varios MHz (Mbps).

Durante la transferencia SPI, los datos son transmitidos y recibidos simultáneamente por cada una de las líneas MISO y MOSI. Esto ocurre porque los datos entrantes en el Esclavo avanzan en un registro interno de forma que cuando se recibe el byte entero, los datos salen por la línea SDO (Ver la figura 1.31). La línea SDO en el esclavo corresponde a la línea MISO comentada anteriormente, mientras que la SDI en el esclavo corresponde a la línea MOSI.

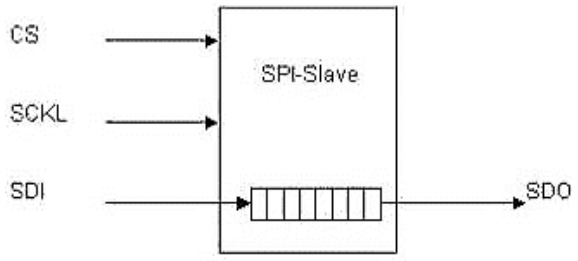

Figura 1.31: Desplazamiento de bits en Esclavo SPI.
Fuente [9]

En la figura 1.32 se puede ver el formato de transferencia del bus SPI.

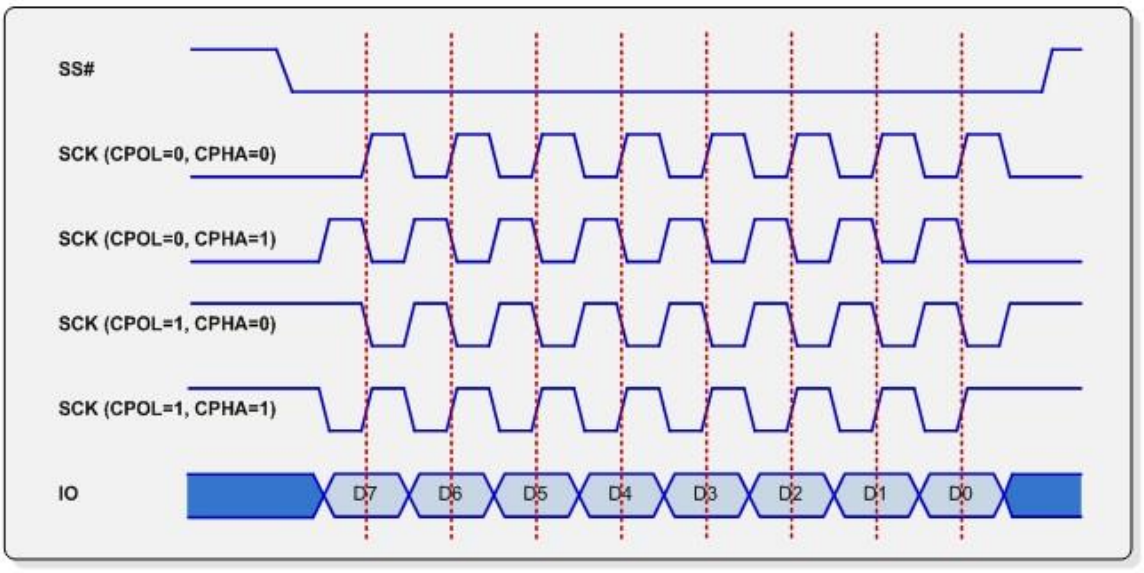

Figura 1.32: Ejemplo de comunicación SPI.
Fuente [9]

1.6. PIC (PERIPHERAL INTERFACE CONTROLLER).

Los PIC son una familia de microcontroladores (μC) fabricados por Microchip Tecnhnology Inc. Para el control de todos los procesos se emplea este tipo de μC debido a su bajo precio, sencillo manejo y programación, y a la cantidad de documentación y usuarios que hay detrás de ellos. Aunque no son los μC que más prestaciones ofrecen, sus características se ajustan perfectamente a nuestro proyecto.

Existe una gran cantidad de modelos de PIC con características y prestaciones diferentes. Esto hace que el desarrollador pueda escoger el modelo que más se ajuste a sus necesidades.

Para que el PIC pueda realizar sus funciones lo primero que se ha de hacer es programarlo, es decir, hemos de escribir un programa que contenga los procesos que el PIC deba ejecutar. Este programa se puede escribir en varios lenguajes de programación, pero los más utilizados son el 'Assembler' (ensamblador) y el C. Aún

faltaría un último paso para que el PIC pueda entender lo que hemos escrito. Este último paso es traducir este programa a lenguaje máquina (1's y 0's). Gracias a los compiladores este proceso es bastante directo. Se pretende utilizar la programación en C y el compilador a utilizar es el proporcionado por la empresa CCS (Custom Computer Services, Inc.).

Como todo μC, un PIC se puede dividir en diferentes bloques (Ver la figura 1.33);

Reloj: Todos los PIC disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia que se utiliza para sincronizar todas las operaciones del sistema. El PIC tiene un oscilador interno incorporado, y en función de la velocidad de trabajo con la que queramos trabajar utilizaremos este oscilador (velocidades más bajas) o un oscilador externo (velocidades altas) como por ejemplo cristal de cuarzo, resonador cerámico[11] o una red R-C. Aumentar la frecuencia del reloj implica disminuir el tiempo de ejecución de las instrucciones, pero lleva aparejado un incremento de la temperatura.

[11] Cristal de cuarzo de baja frecuencia (800kHz).

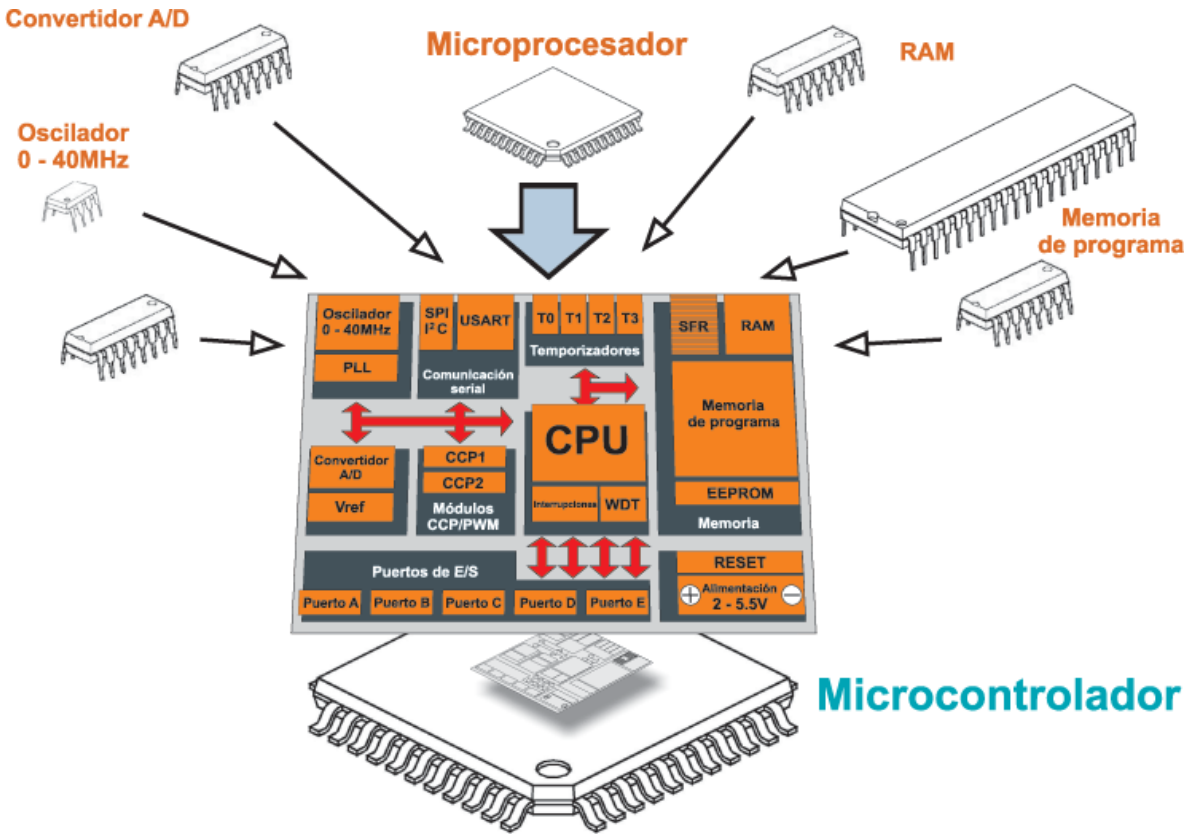

Figura 1.33: Componentes que configuran un microcontrolador.
Fuente [12]

- **I/O (In/Out)**: La mayoría de los pines que posee un PIC son de I/O y se destinan a proporcionar el soporte a las señales de entrada, salida y de control.

- **CPU**: Es el elemento que interpreta las instrucciones y procesa los datos en los programas del PIC.

- **Memoria de datos**: Los datos que manejan los programas varían continuamente, y esto exige que la memoria que los contiene debe ser de escritura y de lectura, por lo que la memoria RAM (Random Access Memory) es la más adecuada, aunque sea volátil. También se dispone de una memoria de lectura y escritura no volátil, del tipo EEPROM (Electrically-Erasable Programmable Read-Only Memory). De esta forma, un corte en el suministro

de la alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa.

- **Memoria de programas**: El PIC está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. Como éste siempre es el mismo, debe estar grabado de forma permanente.
  Existen varios tipos de memoria adecuados para soportar estas funciones, de las cuales en los PIC se utilizan la ROM (Read-Only Memory), OTP (One-Time Programmable) y Flash.

- **Periféricos**: Se llama periféricos a todas aquellas unidades a través de las cuales el PIC se comunica con el mundo exterior. En los PIC podemos encontrar ADC (Analog to Digital Converter), comparadores, temporizadores, y periféricos destinados a las comunicaciones como los siguientes:

  - ✓ EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter): Interfaz entrada salida serie. Con este módulo podemos transformar los datos en serie a paralelo y al revés. De esta forma se pueden adaptar los datos con los que trabaja el PIC (en paralelo) con los que le pueden llegar a través de este puerto (serie). Es el utilizado por el protocolo RS232.

  - ✓ USB (Universal Serial Bus): Permite operar con el protocolo USB.

  - ✓ MSSP (Master Synchronous Serial Port): Es una interfaz serie integrada en el PIC diseñada para comunicarse con otros periféricos o µC. Permite operar con los protocolos I2C y SPI.

  - ✓ CAN (Controller Area Network): Permite operar con el protocolo CAN.

  - ✓ PSP (Parallel Slave Port): Interfaz para conectar dos µC mediante niveles TTL (Transistor-Transistor Logic).

- ✓ CCP/ECCP (Enhanced Capture/Compare/PWM): Módulo que se puede utilizar como comparador, como capturador o como PWM (Pulse-Width Modulation).

Para finalizar creemos importante destacar que los proyectos en los que se suelen trabajar con PIC's pueden ser muy variados, por ejemplo la automoción, la industria, informática, comunicaciones, etc. (Ver la figura 1.34).

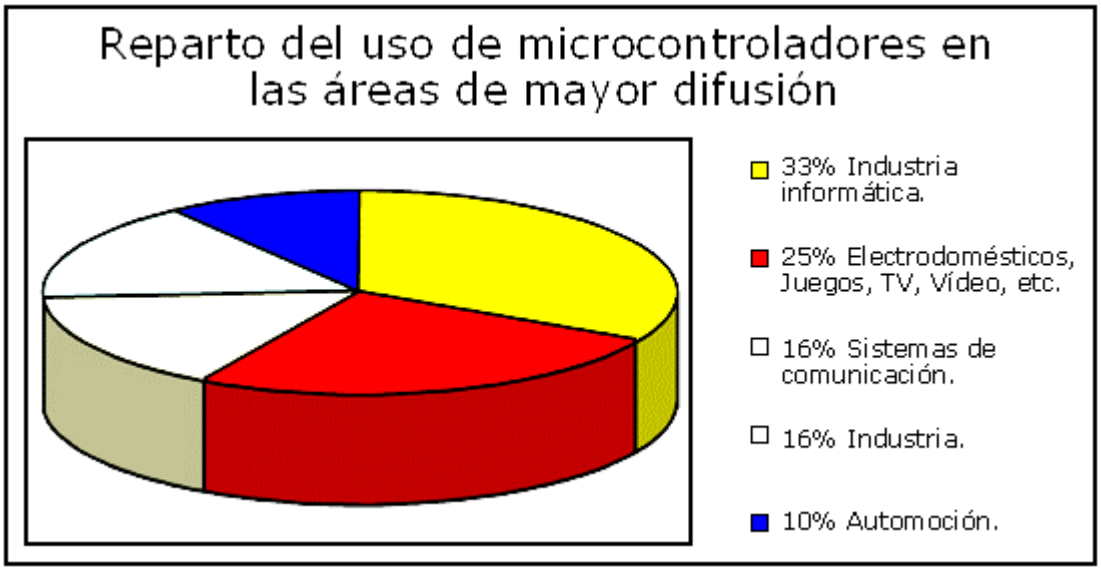

Figura 1.34: Mercado mundial de los PIC's.
Fuente [13]

1.6.1. MICROCONTROLADOR PIC18F4685.

Una vez vistas las características generales de los PIC, nos hace falta conocer las especificaciones del PIC a utilizar. El PIC elegido para es el PIC18F4685 ya que puede soportar los 3 módulos necesarios (EUSART, MSSP y CAN) para trabajar con

los 4 protocolos (RS232, SPI, I2C y CAN). Este PIC tiene 40 pines que se distribuyen como se muestra en la figura 1.35.

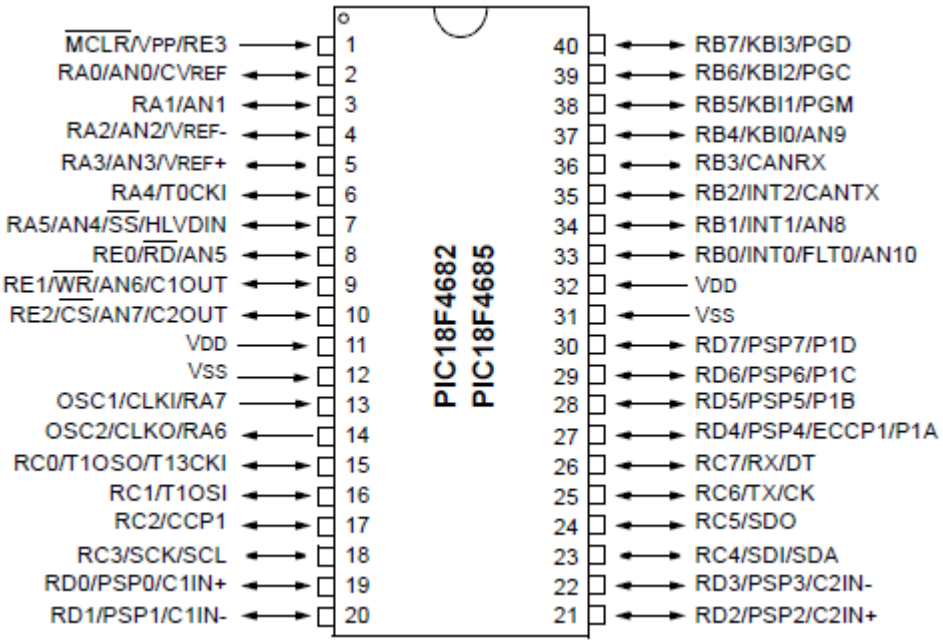

Figura 1.35: Diagrama de pines del PIC 18F4685.
Fuente [15]

Las características más importantes son las siguientes:

- **Reloj**: Ofrece varias opciones de configuración de la frecuencia de oscilación, permitiendo al usuario escoger según se adapte a sus necesidades:
  - Utilizando un cristal o un resonador cerámico y conectándolos a los pines OSC1 y OSC2 del PIC. Para la elección de la frecuencia de oscilación, el fabricante nos ofrece unas tablas (Ver la tabla 1.5); Los condensadores elegidos van del pin OSC1 u OSC2 a 0V (GND). Se

pretende trabajar con una frecuencia de oscilación 40 MHz usando la tecnología PLL[12] disponible, lo que permite usar un cristal de 10 Mhz.

- Utilizando relojes externos, ofreciendo la opción de utilizar dos pines del PIC (OSC1 y OSC2) o sólo un pin (OSC1).

- Utilizando un oscilador externo RC con la misma opción de configuración de pines que en el caso anterior. Un oscilador RC está formado por un condensador no polarizado y una resistencia. Este tipo de oscilador proporciona una estabilidad mediocre en la frecuencia generada y podrá ser utilizado para aquellos proyectos que no requieran precisión.

- Utilizando un oscilador interno. Este método se suele utilizar en los casos que se quieran aprovechar los pines OSC1 y OSC2 como I/O.

- Utilizando un multiplicador de frecuencia PLL. Gracias al PLL y utilizando el oscilador interno, el usuario puede disponer de una selección de frecuencias entre 31kHz y 32MHz.

- I/O: En este PIC hay 5 puertos diferentes (A, B, C, D y E). Cada puerto tiene tres registros para sus operaciones que son el TRIS (registro de dirección de datos), el PORT (el que lee el nivel de tensión que hay en el pin) y el LAT (utilizado en operaciones lectura-modificación-escritura del valor que el pin I/O está leyendo).

- Memoria de datos: Tiene 3328 bytes de memoria RAM y 1024 bytes de EEPROM.

- Memoria de programa: Tiene 96 kbytes de memoria Flash.

[12] El lazo de amarre de fase (PLL, por sus siglas en inglés) es un sistema de retroalimentación. El propósito en este caso es que la frecuencia se cuatro veces (4X) la frecuencia del cristal empleado.

- Periféricos: En el PIC18F4685 podemos encontrar 11 ADC de 10 bits, dos módulos CCP/ECCP, MSSP para I2C y SPI, EUSART, dos comparadores, 4 temporizadores (uno de 8 bits y tres de 16 bits) y un módulo CAN.

Tabla 1.5: Configuración de condensadores según el reloj.
Fuente [15]

Resonadores Cerámicos				Osciladores de cristal			
Modo	**Frecuencia**	**OSC1**	**OSC2**	**Modo**	**Frecuencia**	**OSC1**	**OSC2**
	455kHz	56pF	56pF		32kHz	33pF	33pF
	2MHz				200kHz		
	4MHz	47pF	47pF	**LP**		15pF	15pF
XT[13]		33pF	33pF		1MHz	33pF	33pF
	8MHz	27pF	27pF	**XT**	4MHz	27pF	27pF
HS	16MHz	22pF	22pF		4MHz	27pF	27pF
				HS	8MHz	22pF	22pF
					20MHz	15pF	15pF

1.7. MEMORIAS SD.

Las tarjetas SD (Secure Digital) son tarjetas de memoria Flash (NAND) con controlador inteligente incorporado. Derivan de las tarjetas MMC (MultiMediaCard). Son las tarjetas de memoria más utilizadas en el mercado.

El formato Secure Digital incluye cuatro versiones de tarjetas, disponibles en tres tamaños. Las cuatro familias son la original "Standard Capacity" (SDSC), "High-Capacity" (SDHC), "Extended-Capacity" (SDXC) y SDIO, ver la figura 1.36. Los tres tamaños son el original, el mini, y el micro, los que se muestran en la figura

[13] XT corresponde a 'Crystal/Resonator', HS a 'High-Speed Crystal/Resonator' y LP a 'Low-Power Crystal'

1.37. Por medio de adaptadores eléctricamente pasivos es posible utilizar tarjetas en ranuras más grandes, en la figura 1.38 se muestra un adaptador.

Figura 1.36: Versiones de memorias SD.
Fuente [15]

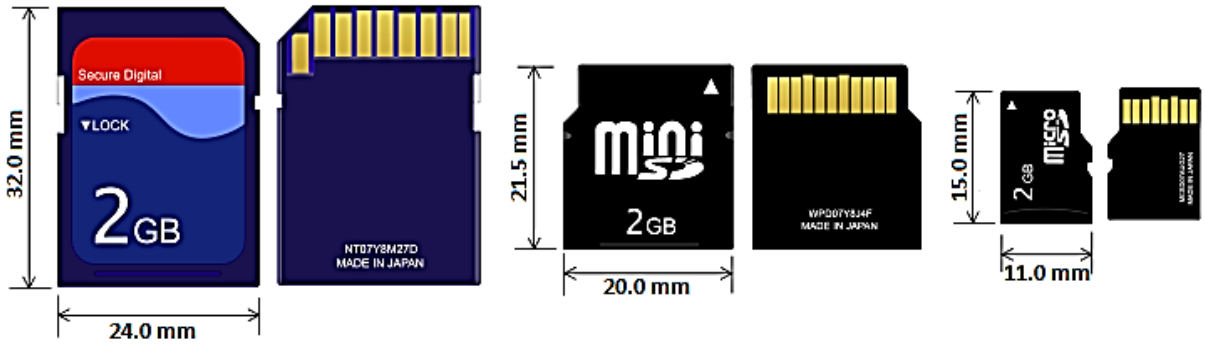

Figura 1.37: Tarjetas SD, mini SD y micro SD, respectivamente.
Fuente [15]

Figura 1.38: Adaptador micro SD.
Fuente [15]

Las memorias SD (original) tienen capacidades que van desde 128MB hasta 2GB, el formato predeterminado es FAT16, funcionan en todos los dispositivos compatibles con SD, SDHC o SDXC.

Las versiones SDHC tienen capacidades que van desde los 4GB hasta 32GB, el formato predeterminado es FAT32. Debido que las tarjetas SDHC funcionan de manera diferente de las tarjetas SD estándar, este nuevo formato **NO** es compatible con dispositivos que soportan sólo tarjetas SD (128MB-2GB). La mayoría de los lectores y dispositivos construidos después de 2008 deben ser compatibles con SDHC. Para asegurarse de la compatibilidad, busque el logotipo SDHC en las tarjetas y los dispositivos de acogida (cámaras, videocámaras, etc.)

Las versiones SDXC tienen capacidades que van desde los 64GB hasta 2TB, el formato predeterminado es exFAT. Porque el SDXC utiliza un sistema de archivo diferente llamado exFAT y funciona de manera diferente de las tarjetas SD estándar, este nuevo formato **NO** es compatible con dispositivos de acogida que soportan sólo SD (128MB a 2GB). La mayoría de los dispositivos de acogida construidos después de 2010 deben ser compatibles SDXC. Para asegurarse de la compatibilidad, busque

el logotipo SDHC en las tarjetas y los dispositivos de acogida (cámaras y videocámaras, etc.).

Lectores de tarjetas internos en portátiles desde 2008 y anteriores NO admitan tarjetas SDXC. Las tarjetas SDXC funcionarán en lectores compatibles con SDHC (no con lectores SD) si el sistema operativo soporta exFAT.

La versión SDIO no es una tarjeta física, más bien, se refiere que el dispositivo lector es de este tipo (SDIO), esto le da la capacidad al dispositivo huésped (host) la capacidad de utilizar el lector SD como una interface a la que se le pueden conectar dispositivos especiales que traen un conector en forma de memoria SD, como el de la imagen 1.39.

Figura 1.39: Una cámara con interface SDIO.
Fuente [15]

Los dispositivos que soportan SDIO (típicamente PDA, pero cada vez más ordenadores portátiles y teléfonos móviles) pueden usar pequeños dispositivos diseñados para las dimensiones SD, como receptores GPS, Wi-Fi o adaptadores Bluetooth, módems, lectores de códigos de barras, adaptadores IrDA, sintonizadores de radio FM, lectores de RFID o cámaras digitales acoplables.

Se han propuesto otros dispositivos, pero todavía no se han implementado, como los adaptadores serie RS-232, sintonizadores de TV, escáneres de huella dactilar, adaptadores maestro/esclavo de SDIO a USB (que permitirían que un dispositivo de mano equipado con SDIO utilizara periféricos USB o interfaz a PC), lectores de bandas magnéticas, transmisores-receptores de Bluetooth/Wi-Fi/GPS, adaptadores Ethernet y adaptadores de módems celulares (PCS, CDPD, GSM, etc.).

1.7.1. GRADOS DE CLASE DE VELOCIDAD.

Los grados de clase de velocidad define una velocidad mínima garantizada de tarjetas SDHC/SDXC. El índice de velocidad de clase es importante para el modo de vídeo HD o videocámaras, donde el dispositivo está ahorrando realmente un flujo de datos continuo. La resolución y el formato de vídeo determinan la cantidad de datos de flujo continuo. Usted debe consultar el manual del usuario de la cámara para un mínimo de requisitos de velocidad de clase para modos de vídeo HD.

Actualmente las velocidades mínimas garantizadas de transferencia que aseguran las tarjetas han sido estandarizadas con las nomenclaturas que se muestran en la tabla siguiente:

Tabla 1.6: Clase de velocidad (Bus SD).
Fuente [15]

CLASE	VELOCIDAD
 Clase 2	2 MB/s
 Clase 4	4 MB/s

Clase 6	6 MB/s
Clase 10	10 MB/s
UHS Clase 1	10 MB/s
UHS Clase 3	30 MB/s

1.7.2. COMUNICACIÓN CON EL HOST.

El Host (microprocesador) se comunica con la tarjeta mediante una interfaz eléctrica de 9 pines:

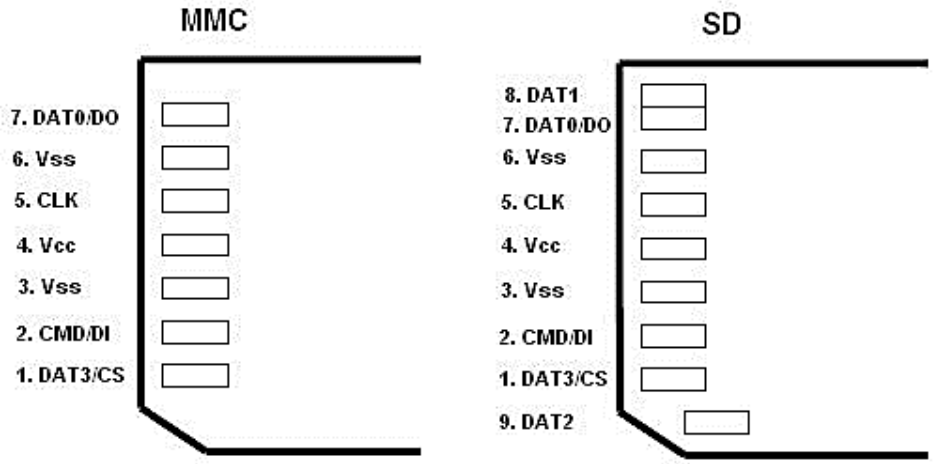

Figura 1.40: Pinout de la tarjeta SD/MMC.
Fuente [16]

Las tarjetas soportan 2 modos de comunicación: Modo SD y Modo SPI. El Modo SD es el modo nativo, y permite mayor velocidad de transferencia que el modo SPI. Las ventajas de este último modo son la simplicidad y la disponibilidad

del periférico de comunicaciones SPI en la mayoría de los microcontroladores. Vamos a ver sólo el modo SPI.

1.7.3. MODO SPI.

1.7.3.1. PINOUT.

La interfaz consta de 4 pines:

Tabla 1.7: Asignación de pinout en el modo SPI.
Fuente [17]

Pin	Nombre	Funcionalidad
1	CS	Chip Select (activo en bajo)
2	MOSI	Entrada de datos
3	GND	Tierra
4	VCC	Alimentación
5	SCK	Reloj
6	GND	Tierra
7	MISO	Salida de datos
8 y 9	RSV	Reservados, conectar pullups

Las tarjetas se alimentan en general a una tensión entre 2.7 y 3.3 voltios, las entradas y salidas usan tecnología LVCMOS (Low Voltage Complementary Metal Oxide Semiconductor, semiconductor de óxido metálico complementario de bajo voltaje.

1.7.3.2. COMUNICACIÓN.

La transferencia (full duplex) se realiza de a bytes, alineados con la señal de reloj, MSb primero. La figura muestra el diagrama de tiempos, visto desde el host (microcontrolador).

Según muestra la figura 1.41, la tarjeta y el host muestrearán las líneas de datos en el flanco ascendente del reloj (clock).

1.7.3.3. VELOCIDAD DE RELOJ.

En general, la tarjeta SD acepta velocidades de reloj desde 0Hz hasta 25MHz. Sin embargo, durante la inicialización, la línea CLK debe estar detenida o entre 100KHz y 400KHz, esto es necesario solo para compatibilidad con tarjetas MMC, por experiencia personal y por la presentada por terceros esto no es necesario para las tarjetas SD.

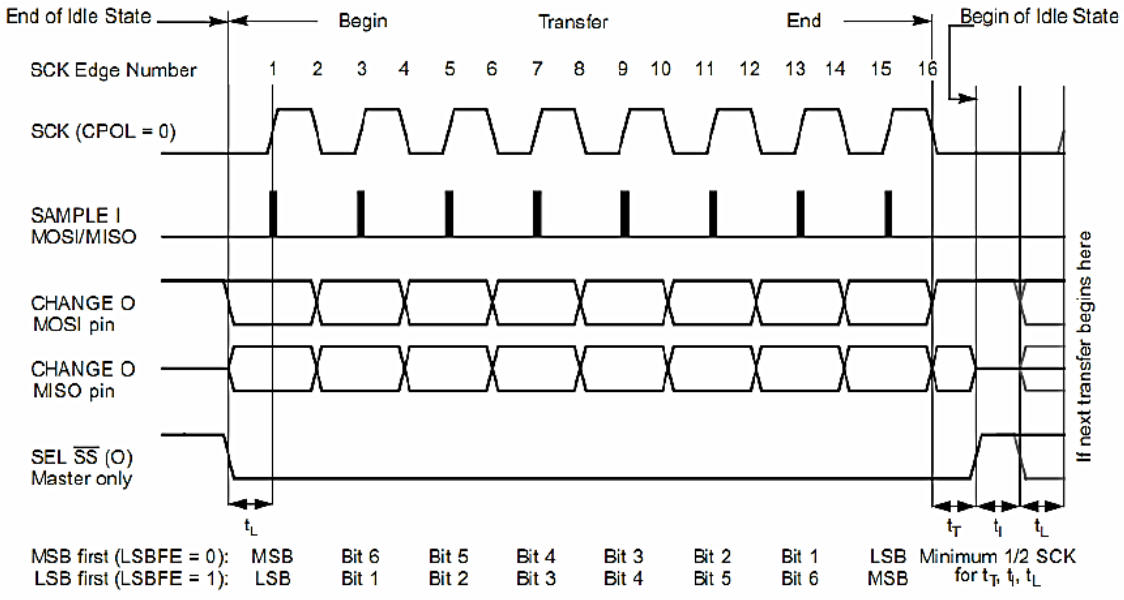

Figura 1.41: Diagrama de tiempos de transmisión y recepción de tarjeta SD en modo SPI. Fuente [17]

1.7.3.4. PROTOCOLO.

Como ya se ha dicho, la transferencia de datos se hace de a bytes. La comunicación con la tarjeta se basa en comandos de 6 bytes, y respuestas asociadas.

La estructura de un comando se muestra en la figura 1.42.

Byte 1				Bytes 2—5		Byte 6	
7	6	5	0	31	0	7	0
0	1	Command		Command Argument		CRC	1

Figura 1.42: Formato de comandos.
Fuente [16]

- **Byte 1:**
  - Bit 7 (MSB) siempre 0.
  - Bit 6 siempre 1.
  - Bit 5 a bit 0 (LSB) comando.

- **Byte 2 a 5:**
  - Bit 31 (MSB) a bit 0 (LSB) argumento, big endian.

- **Byte 6 (LSB):**
  - Bit 7(MSB) a bit 1, CRC del comando.
  - Bit 0(LSB) siempre 1.

Luego de la recepción de un comando, la tarjeta **debe** responder. Hay varios tipos de respuesta:

- R1 (respuesta a la mayoría de comandos).
- R1b (Repuesta idéntica a R1 + algunos bytes de "busy").
- R2 (respuesta al comando SEND_STATUS).
- R3 (respuesta a la lectura del registro OCR).
- Data response (respuesta a la lectura de un sector).
- Write response (respuesta a la escritura de un sector).

Respuesta R1.

Respuesta a la mayoría de comandos, consta de un byte y se muestra en la figura 1.43, un '1' indica que ha habido un error.

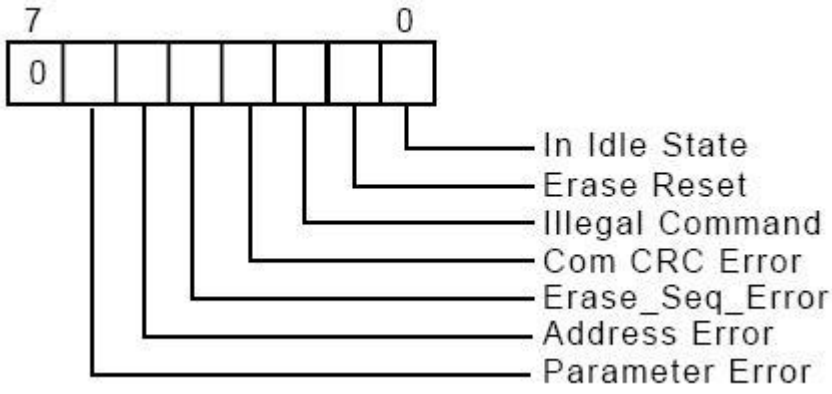

Figura 1.43: Byte de respuesta R1.
Fuente [18]

Respuesta R1b.

La imagen 1.44 muestra la respuesta R1b, es idéntica a R1, solo que luego de recibir la respuesta hay que esperar condición de memoria desocupada en donde la línea SDO de la memoria se lleva a nivel de 0 lógico.

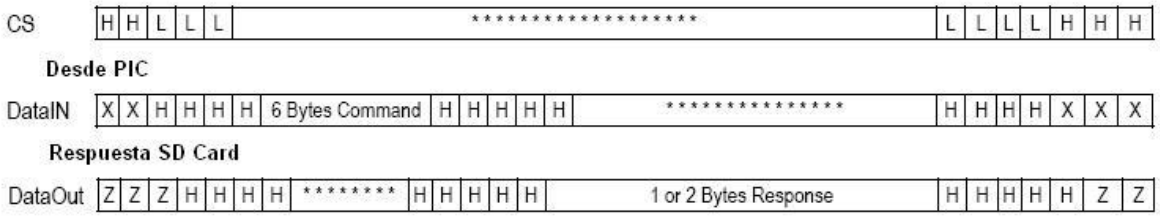

Figura 1.44: Respuesta R1b.
Fuente [16]

Respuesta de lectura de datos.

Dependiendo del éxito de la operación de lectura, la tarjeta responderá de dos formas diferentes:

1. Lectura exitosa.

   Bloque de n=4 a n=515 bytes de longitud:

   - **Byte 1** (MSB). Siempre 0xFE.
   - **Byte 2 a n-2.** Datos.
   - **Byte n-1 y n**. CRC de los datos.

2. Error en el proceso.

   Un solo byte, con valor dependiente del error:

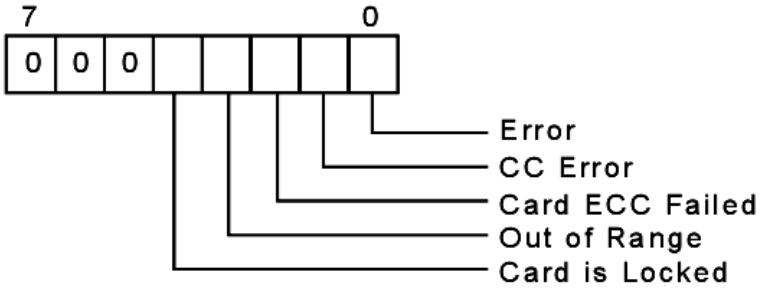

Figura 1.45: Byte de respuesta por error de lectura de datos.
Fuente [17]

Respuesta de escritura de datos.

Luego de enviar un bloque para ser escrito en la tarjeta, ésta devuelve una respuesta, indicando el estado de la operación de escritura:

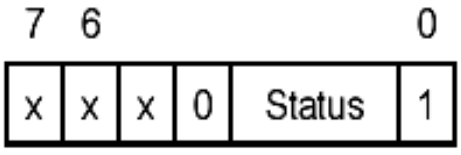

Figura 1.46: Respuesta del estado de la escritura de datos.
Fuente [17]

- Estado = ‘010’ **->** Datos aceptados.
- Estado = ‘101’ **->** Datos rechazados debido a error en CRC.
- Estado = ‘110’ **->** Datos rechazados debido a un error de escritura.

El controlador de la tarjeta acepta varios comandos. En la tabla 1.7 se exponen algunos importantes. Se obvian comandos de lectura/escritura multibloque, etc.

Tabla 1.8: Algunos comandos utilizados para el manejo de tarjetas SD. Fuente [17]

Command Index	Argument	Response	Abbreviation	Description
CMD0	None	R1	GO_IDLE_STATE	Software reset.
CMD1	None	R1	SEND_OP_COND	Initiate initialization process.
ACMD41	None	R1	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD9	None	R1	SEND_CSD	Read CSD register.
CMD17	Address[31:0]	R1 + DATA	READ_SINGLE_BLOCK	Read a block.
CMD24	Address[31:0]	R1	WRITE_BLOCK	Write a block.
CMD55	None	R1	APP_CMD	Leading command of ACMD<n> command.
CMD58	None	R3	READ_OCR	Read OCR.

1.8. RELOJ DE TIEMPO REAL.

Un reloj en tiempo real (en inglés, real-time clock, RTC), es un reloj de un ordenador, incluido en un circuito integrado, que mantiene la hora actual. Aunque el término normalmente se refiere a dispositivos en ordenadores personales, servidores y sistemas embebidos, los RTCs están presentes en la mayoría de los aparatos electrónicos que necesitan guardar el tiempo exacto.

El término se usa para evitar la confusión con los relojes hardware ordinario que sólo son señales que dirigen circuitos digitales, y no cuentan el tiempo en unidades

humanas. Los RTC no deben ser confundidos con la computación en tiempo real (en inglés, real-time computing), que comparte su acrónimo de tres letras, pero que no se refiere directamente al tiempo del día.

Aunque controlar el tiempo puede hacerse sin un RTC, usar uno tiene beneficios:

- Bajo consumo de energía (importante cuando está funcionando con una pila).
- Libera de trabajo al sistema principal para que pueda dedicarse a tareas más críticas.
- Algunas veces más preciso que otros métodos.

Los RTCs a menudo tienen una fuente de alimentación alternativa, por lo que pueden seguir midiendo el tiempo mientras la fuente de alimentación principal está apagada o no está disponible. Esta fuente de alimentación alternativa es normalmente una batería de litio en los sistemas antiguos, pero algunos sistemas nuevos usan un supercapacitor, porque son recargables y pueden ser soldados. La fuente de alimentación alternativa también puede suministrar energía a una memoria no volátil.

La mayoría de los RTCs usan un oscilador de cristal, pero algunos usan la frecuencia de la fuente de alimentación. En muchos casos la frecuencia del oscilador es 32.768 kHz. Ésta es la misma frecuencia usada en los relojes de cuarzo, y por las mismas razones, que la frecuencia es exactamente 2^{15} ciclos por segundo, que es un ratio muy práctico para usar con circuitos de contadores binarios simples.

Muchos fabricantes de circuitos integrados fabrican RTCs, por ejemplo Intersil, Maxim, Philips, Texas Instruments y STMicroelectronics. El RTC fue introducido en los PC compatibles por IBM PC/AT en 1984, cuando usó un RTC MC146818. Posteriormente Dallas fabricó RTCs compatibles que fueron muy usados en

ordenadores personales viejos, y que se pueden encontrar fácilmente en sus placas base por su distintiva batería negra y por su logo serigrafiado. En los sistemas nuevos el RTC está integrado en chip southbridge.

Algunos microcontroladores tienen reloj en tiempo real incorporado, generalmente sólo unas más de otras características y periféricos.

La figura 1.47 muestra un circuito integrado RTC que posee una interfaz I²C y en la figura 1.48 se muestra RTC con interfaz PSI.

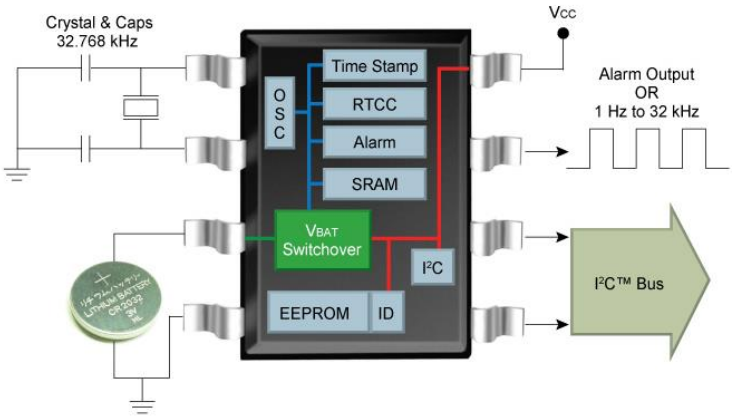

Figura 1.47: IC RTC con interfaz I2C.
Fuente [20]

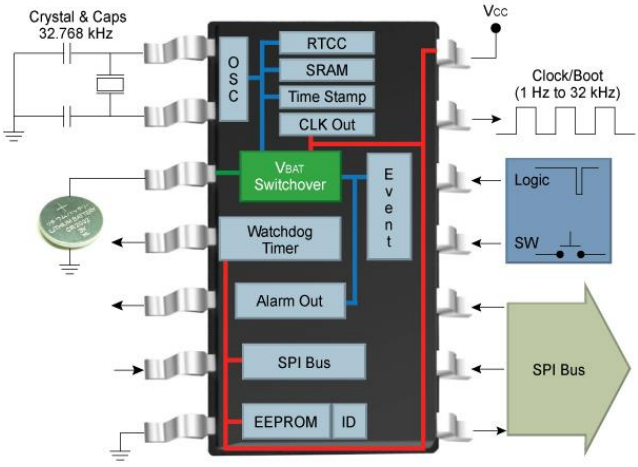

Figura 1.48: IC RTC con interfaz PSI.
Fuente [20]

CAPÍTULO II.

DISEÑO E IMPLEMENTACIÓN DEL HARDWARE.

Introducción.

En este capítulo se muestra el diseño de la etapa de hardware que se realiza para llevar a cabo el proceso de medición de energía. Se describe cada uno de los componentes usados y su función en cada módulo.

El principio de funcionamiento básico de un medidor de estado sólido está conformado por cuatro etapas principales que son:

- Acondicionamiento de señales.
- Adquisición, filtrado, procesamiento de señales digitales y almacenamiento.
- Visualización.
- Transmisión de las mediciones.

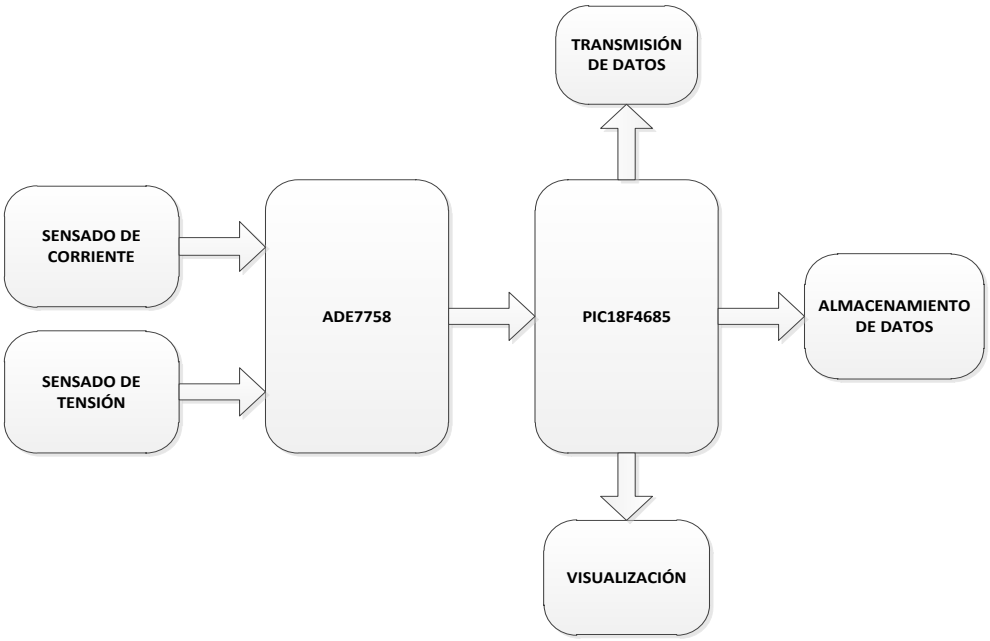

Figura 2.1: Diagrama de bloques del funcionamiento del medidor de energía.

Fuente [Autor]

En los medidores de estado sólido, las señales analógicas de tensión y corriente son adquiridas y digitalizadas tomando muestras y convirtiendo estas muestras en un registro. Una vez digitalizadas, los valores de las muestras son utilizadas para calcular los parámetros requeridos para evaluar las potencias y energías del sistema. Estas métricas son almacenadas en memoria y están disponibles para su utilización; este proceso corresponde a la etapa de procesamiento de señales.

Las estimaciones realizadas son visualizadas en una pantalla de cristal líquido (LCD) o leídas de la memoria del equipo por medio del concentrador haciendo uso del bus CAN y su posterior envío por medio de un puerto Ethernet a una base de datos a la que se puede tener acceso utilizando una interfaz en la PC.

2.1. ACONDICIONAMIENTO DE SEÑALES.

Este módulo consta de una etapa para adquirir las señales de tensión y corriente de la red eléctrica trifásica y acondicionar estas señales a niveles adecuados.

Las señales analógicas de salida son proporcionales a las señales de entrada y tienen una amplitud máxima de 0.5V cuando en la entrada hay plena escala. En total existen seis señales de salida, de las cuales tres corresponden a tensión (Va, Vb, Vc) y tres a corriente (IA, IB, IC). La corriente de neutro no es utilizada por los circuitos integrados para sus cálculos de potencia, por lo tanto no es necesaria su medida. La figura 2.2 representa el acondicionamiento de la señal de corriente aplicado, los valores de corriente I1 y I2 puede variar dependiendo del sensor que se emplee, además la salida del sensor puede ser también una señal de tensión, los valores que se muestran corresponden al sensor que se está usando.

El acondicionamiento de la señal de tensión se muestra en la figura 2.3.

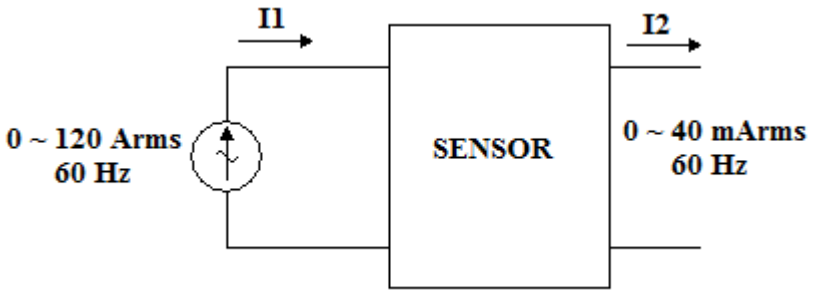

Figura 2.2: Acondicionamiento de la señal de corriente.
Fuente [Autor]

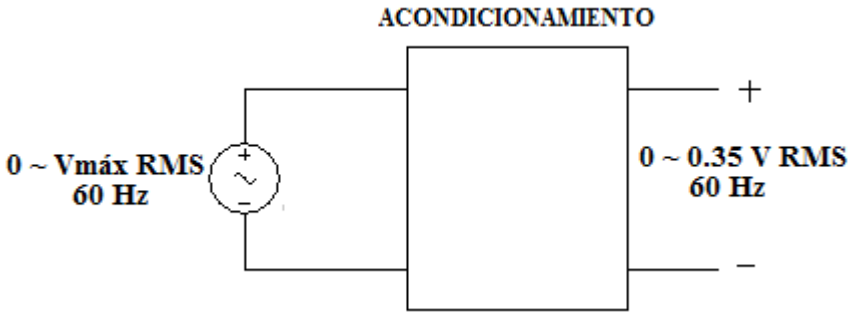

Figura 2.3: Acondicionamiento de la señal de tensión.
Fuente [Autor]

2.2. SENSADO DE CORRIENTE.

Para sensar corriente se utiliza un transformador de corriente (SCT-016) de la empresa YHDC. Este transformador tiene un núcleo de ferrita, en la figura 2.4 se aprecian las dimensiones físicas y en la tabla 2.1 se encuentran los parámetros básicos del transformador de corriente.

La salida del transformador es una señal de corriente, pero el circuito integrado ADE7758 procesa señales de tensión, es por ello que es necesario hacer una conversión de corriente a voltaje, lo que se logra implementando el circuito de la

figura 2.5, este diseño toma en cuenta que la corriente máxima capaz de medir el sistema es de 120 Arms y a plena señal esta corriente debe generar 0.35 Vrms, que corresponde al voltaje máximo permisible por circuito integrado ADE7758.

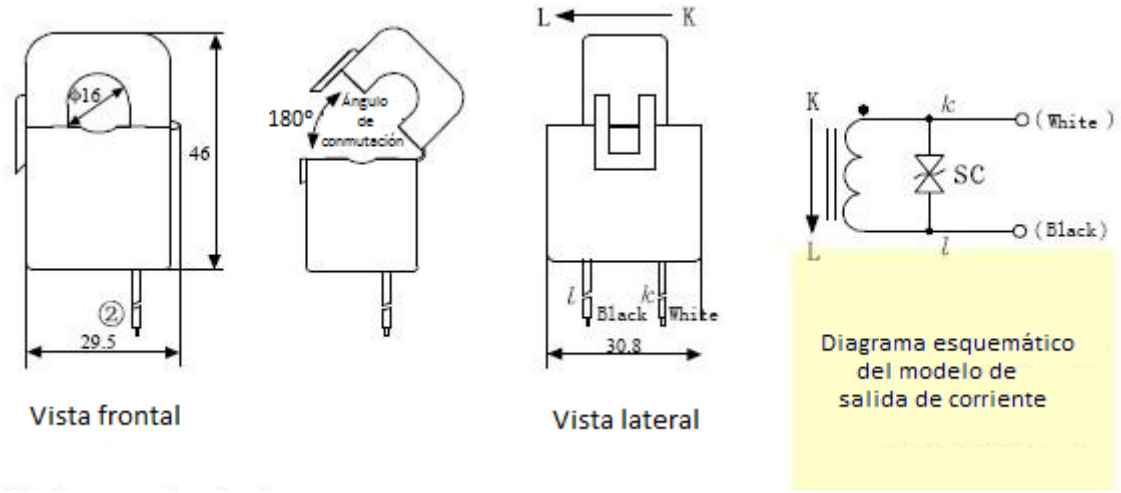

Figura 2.4: Dimensiones físicas del transformador de corriente (en mm). Fuente [19]

Tabla 2.1: Hoja de parámetros básicos.
Fuente [19]

Rango de corriente de entrada	0.01 – 120 Arms (50-60Hz) RL ≤ 10
Corriente máxima	300 Arms continuos
Corriente de salida	40 mA
Linealidad	± 1.0
Frecuencia apropiada	50Hz – 150KHz
Numero de vueltas del secundario(n)	3000 ± 2vueltas
Resistencia del secundario	280Ω ± 20Ω
Alambre principal del secundario	Cable UL-1007 (AWG 22)
Tensión de rigidez dieléctrica (Pri/Sec)	1000 VAC / 1 min
Resistencia de aislamiento (Pri/Sec)	Más de 500 VDC / 100MΩ
Rango de temperatura de trabajo	-20°C ~ +50°C
Temperatura de almacenamiento	-30°C ~ +90°C
Peso	70 g

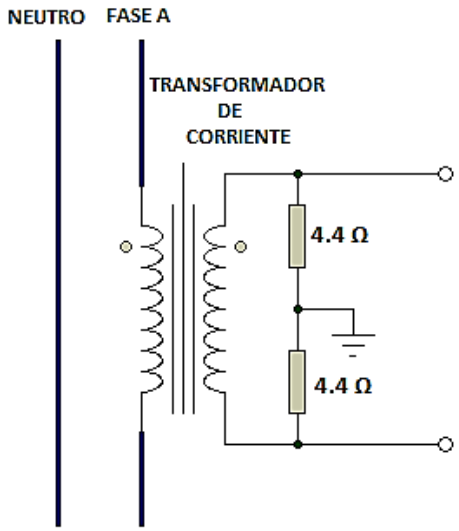

Figura 2.5: Conversión de la señal de corriente a señal de voltaje.
Fuente [Autor]

2.3. SENSADO DE TENSIÓN.

Para cada fase se implementa un divisor de tensión resistivo, figura 2.6, entre las terminales de la fase y tierra analógica del circuito. Esto se realiza con el fin de llevar la señal de entrada a un nivel óptimo para la adecuación de las señales a cada uno de los circuitos integrados para su posterior procesamiento. Específicamente la tensión de salida del divisor tiene una amplitud de 0.35 Vrms cuando en su entrada se aplica una tensión máxima de 480 Vrms.

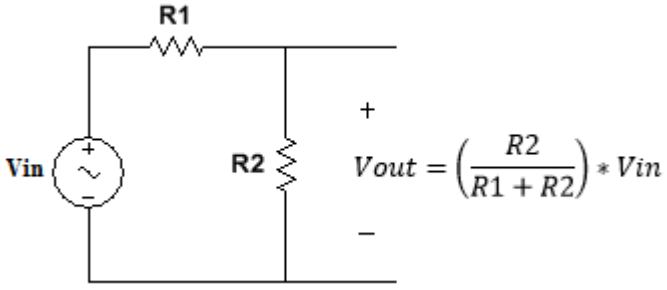

Figura 2.6: Divisor de tensión.
Fuente [Autor]

2.4. FILTRADO DE LAS SEÑALES.

Es necesario una etapa de filtrado de las señales que entran a los circuitos integrados de propósito específico para evitar el efecto de "aliasing". El aliasing es un fenómeno que aparece en todos los sistemas de muestreo. Señales de entrada con componentes de frecuencia más altas que la mitad de la tasa de muestreo de los conversores A/D, distorsionan las muestras de señal a frecuencias por debajo de la mitad de la tasa de muestreo.

Tomando como referencia las especificaciones del fabricante, se implementa un filtro paso bajos en cada una de las entradas analógicas de los conversores. Un simple filtro RC con valores de R=1.6kΩ y C=10nF garantiza una frecuencia de corte de 10 KHz, suficiente para eliminar el aliasing, el circuito de filtrado implementado se muestra en la figura 2.7.

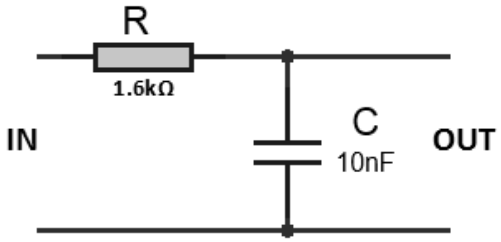

Figura 2.7: Filtro anti-aliasing.
Fuente [Autor]

2.5. ADQUISICIÓN DE SEÑALES DE CORRIENTE Y DE TENSIÓN.

El ADE7758 tiene un total de seis entradas análogas divididas en dos canales: corriente y tensión. El canal de corriente consiste de tres pares de entradas de tensión diferencial: IAP y IAN, IBP y IBN, ICP y ICN. Estas entradas de voltaje diferencial deben tener una amplitud máxima de ±0.5V. Este canal tiene un amplificador de ganancia programable (PGA) con posibles selecciones de ganancia de 1, 2 o 4, permitiendo que el valor de entrada sea ajustado a ±0.5V, ±0.25V ó ±0.125V

respectivamente. Lo anterior se logra escribiendo el valor deseado de ganancia en los bits 3 y 4 del registro GAIN REGISTER.

El canal de tensión tiene tres entradas: VAP, VBP, VCP. Estas entradas deben tener ±0.5V como máximo con respecto a VN (neutro). Al igual que el canal de corriente este canal cuenta con un amplificador programable mediante los bits 5 y 6 del registro GAIN REGISTER.

El nivel de tensión para la máxima conversión o el nivel máximo permisible que se escogió para trabajar es ± 0.5V en los canales de corriente y de tensión. Para obtener el valor deseado en los pines del ADE7758 se utiliza un divisor de tensión resistivo.

En la figura 2.8 se muestra el circuito implementado para la adquisición de la señal de corriente y en la figura 2.9 se muestra el circuito para la señal de tensión, ambos diagramas corresponden a la fase A, pero es el mismo diagrama para las fases B y C.

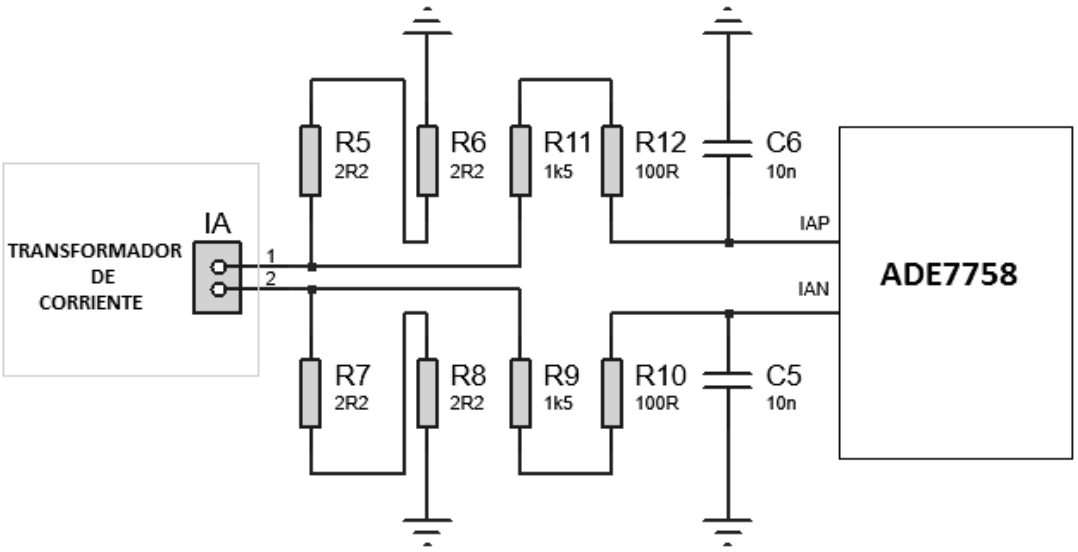

Figura 2.8: Adquisición de la señal de corriente.
Fuente [Autor]

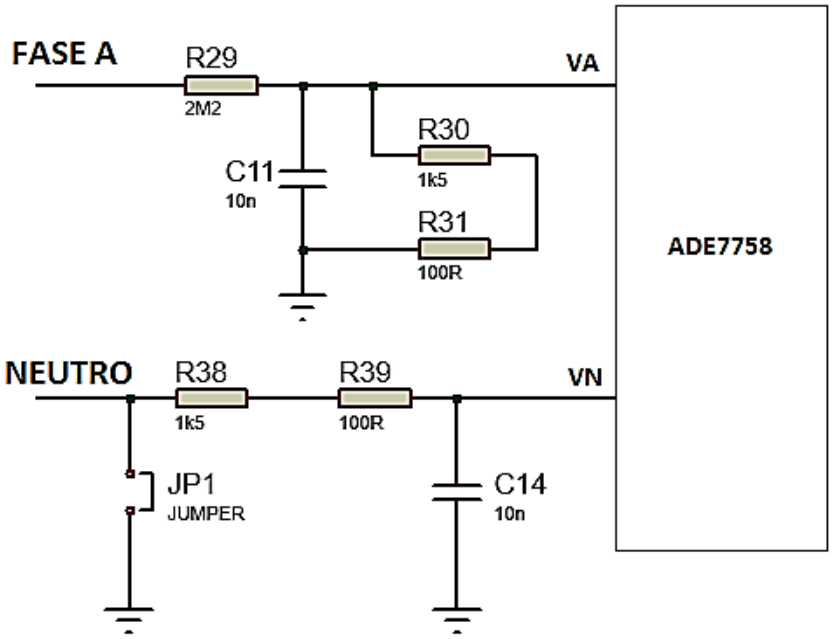

Figura 2.9: Adquisición de la señal tensión.
Fuente [Autor]

2.6. ALMACENAMIENTO DE DATOS.

Para el almacenamiento de datos se hace uso de una tarjeta SD que es operada con el módulo SPI que posee el microcontrolador PIC18F4685, estas tarjetas tienen un bajo precio y una gran capacidad de almacenaje, lo que permite el almacenamiento de varios días de medición de los parámetros eléctricos de interés, sin la necesidad de trasmitirlos a una un servidor de base de datos. Para una fácil gestión de la memoria SD, se ejecuta el formato FAT16, esto conlleva a la limitante que se puede usar una tarjeta SD que no supere los 2GB de capacidad.

El microcontrolador funciona con niveles de voltaje TTL, mientras que la tarjeta SD emplea niveles CMOS, por lo cual se necesita hacer una conversión de niveles voltaje en los pines que son de entrada de la tarjeta SD para proteger de daños la tarjeta, pero no es necesario con los pines de salida, ya que estos niveles de voltaje son

tolerables para el microcontrolador. En la figura 2.10 se muestra el diagrama usado para el control de la tarjeta SD, en la tabla 2.2 se muestran los pines del microcontrolador empleados para esta función.

Tabla 2.2: Asignación de pines en el microcontrolador para la tarjeta SD.
Fuente [Autor]

Función	Pin de puerto	Numero de pin
Salida de dato serial (SDO)	RC5	24
Entrada de dato serial (SDI)	RC4	23
Reloj serial (SCK)	RC3	18
Habilitador (SS_SD)	RC0	15
Detectar tarjeta (CD)	RC1	16
Tarjeta bloqueada (CL)	RC2	17

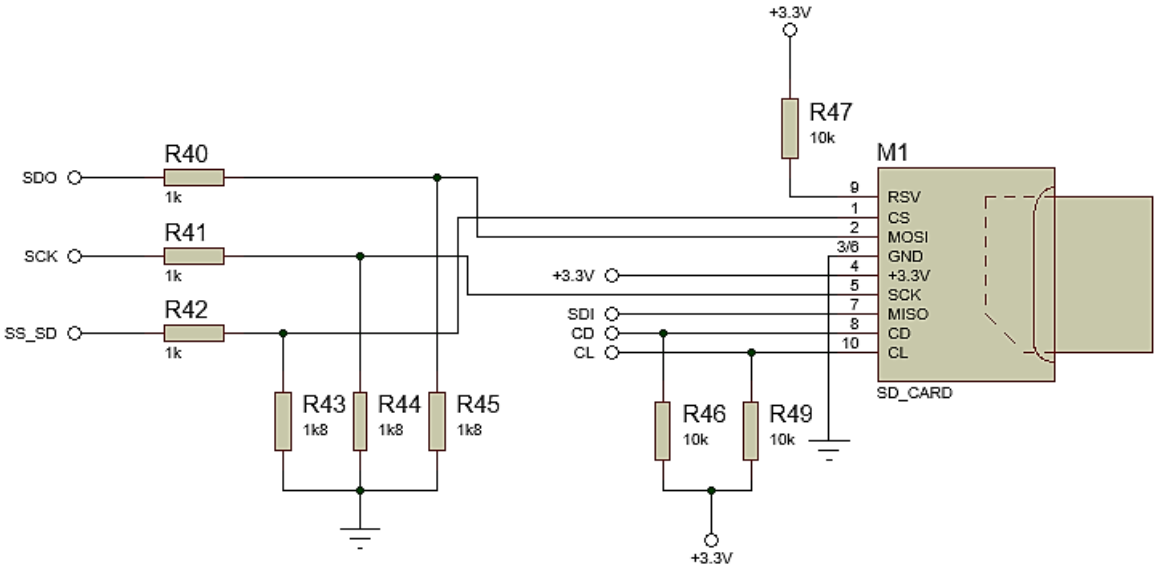

Figura 2.10: Conexión de la rajeta SD.
Fuente [Autor]

2.6.1. FIRMWARE PARA EL MANEJO DE LA MEMORIA SD.

Para poder acceder a la tarjeta SD lo primero por hacer es cambiar el modo de trabajo de la misma, por defecto inicia el modo SD, el inicio de la tarjeta en modo SPI se realiza por medio del procedimiento que aparece en el diagrama de flujo de la figura 2.11.

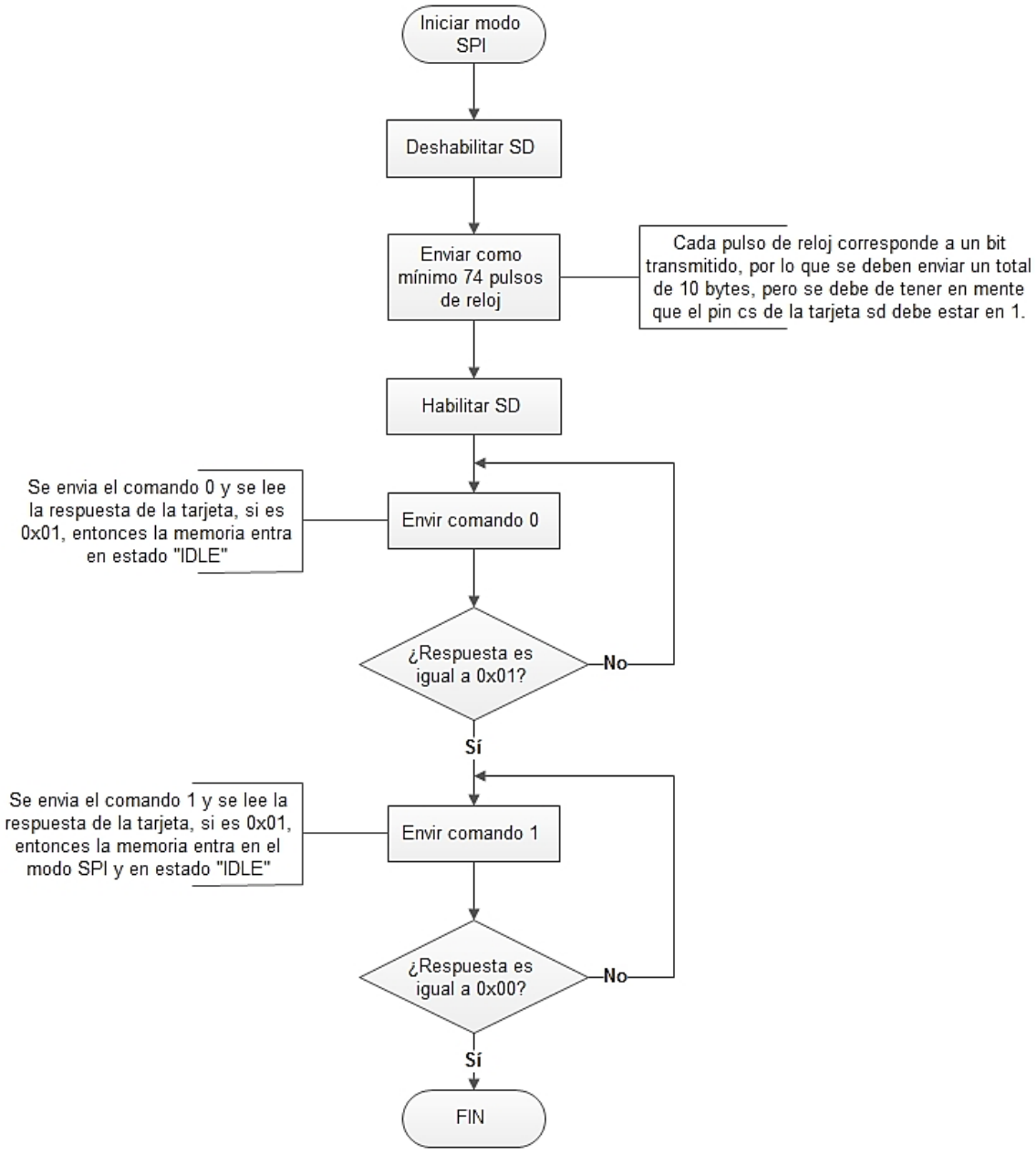

Figura 2.11: Procedimiento para el inicio del modo SPI de la tarjeta SD. Fuente [Autor]

Con la tarjeta SD trabajando en modo SPI, la lectura o escritura se inicia enviando el comando respectivo, dicho comando debe contener la dirección física de inicio del sector a usar.

Para conocer la dirección física del sector a usar se necesita un programa que lea y edite la tarjeta SD, por ejemplo el WinHex. En la figura 2.12 se señalan en rojo los parámetros de importancia que se pueden a preciar al usar el WinHex.

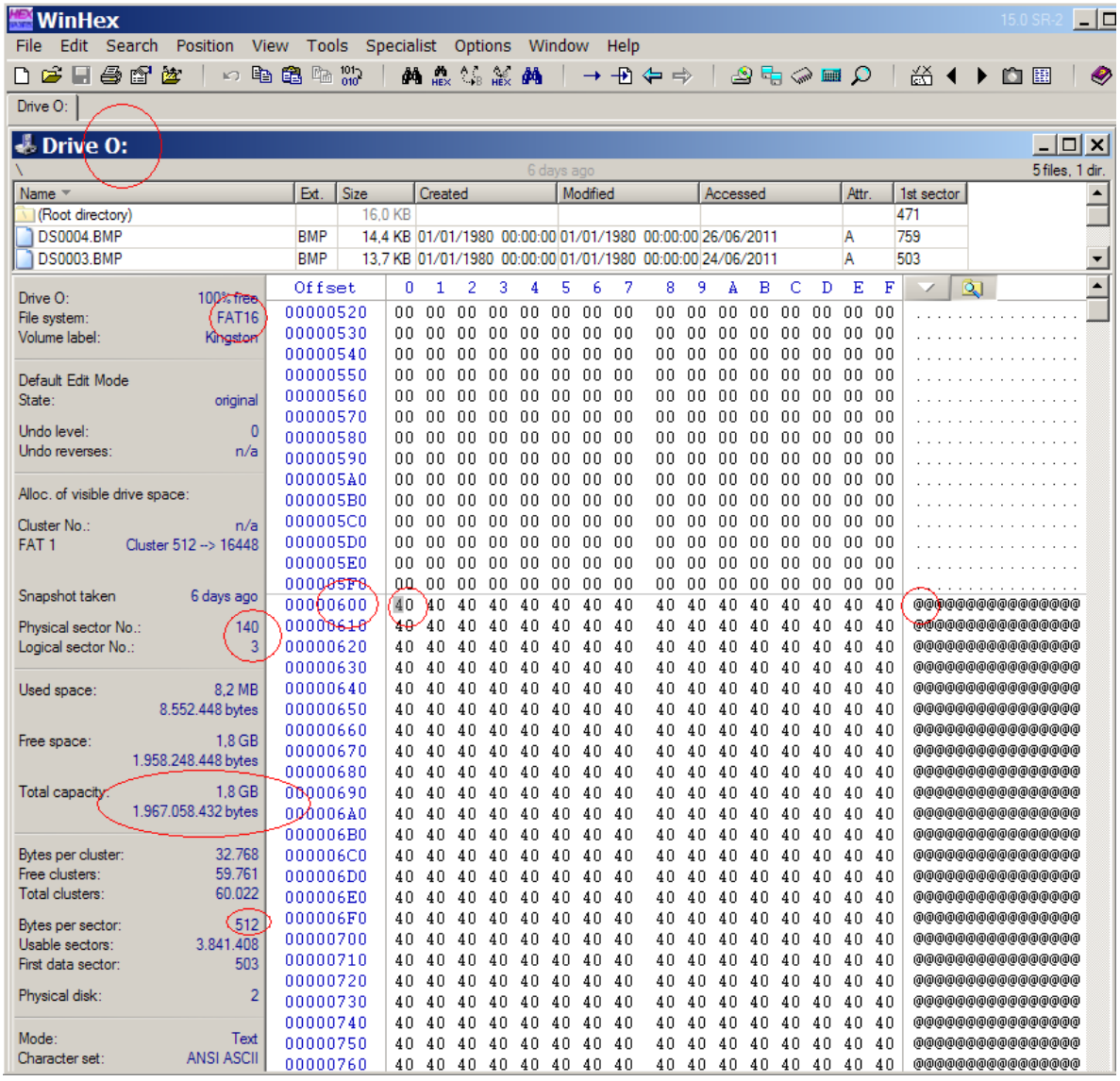

Figura 2.12: Captura de pantalla del programa WinHex.
Fuente [18]

Está conectada en un USB, drive "O", Formateada en FAT16, muestro el sector 3, que comienza en la dirección 600 hexadecimal, tiene 1.8 GB de capacidad útil y casi 2 GB de capacidad total. 512 Bytes por sector.

Lo más importante es donde dice "Sector físico = 140" y "Sector lógico= 3". Los "@" los grabados comienzan en la dirección lógica 0x600. La dirección lógica cero corresponde a la física 137 (Decimales), de manera que en lugar de marcar como inicio de escritura o lectura la dirección 0x600 = sector 3 x 0x200, se deberá calcular (3 + 137) x 512 = 71680, o sea 0x11800.

En una segunda memoria SD de la misma capacidad, el sector lógico cero es 135 físico, de manera que para cada tarjeta se debe mirar el número y hacer los cálculos.

En caso de una mala programación y grabar el sector cero, se pierde la FAT, con el resultado de "Tarjeta imposible de leer" en la PC.

Para formatear la SD, usar el programa oficial de tarjetas SD, SD Formatter.

Para realizar una lectura o escritura en la tarjeta SD se envía el comando respectivo y dentro de dicho comando contener la dirección de inicio de la operación a realizar, los comandos están compuestos por 6 bytes y tienen el formato mostrado en la figura 1.42 de la sección 1.7.

A manera de ejemplo para realizar una lectura de un sector de la tarjeta SD se emplea el comando 17 y se calcula de la siguiente forma: *17 pasado a es 0b10001; se le agrega 01 de los bit 7 y 6 y 0 para el bit 5 más los 5 bit de 10001, quedando 0b01010001, se convierte a hexadecimal quedando 0x51, que es el valor mostrado para comando 17.*

El contenido total del comando 17 para leer el sector lógico 3 se muestra en la tabla 2.3, un procedimiento similar se realiza para el comando 24 empleado para escribir en la tarjeta SD.

Tabla 2.3: Contenido del comando 17.
Fuente [18]

BYTE 1	BYTE 2 - 5	BYTE 6
0x51	0x011800	0xFF

En la figura 2.13 muestra el diagrama de flujo que se debe a seguir para la lectura de un sector de la tarjeta SD y la figura 2.14 muestra el procedimiento para la escritura de un sector de la memoria SD.

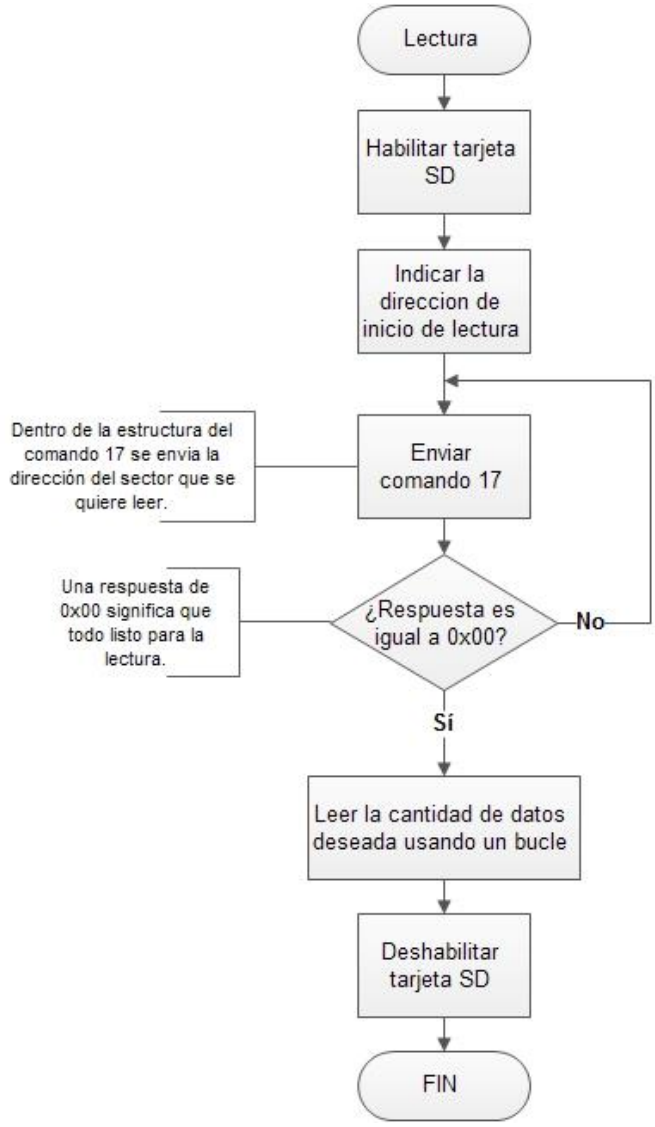

Figura 2.13: Proceso de lectura en la tarjeta SD.
Fuente [Autor]

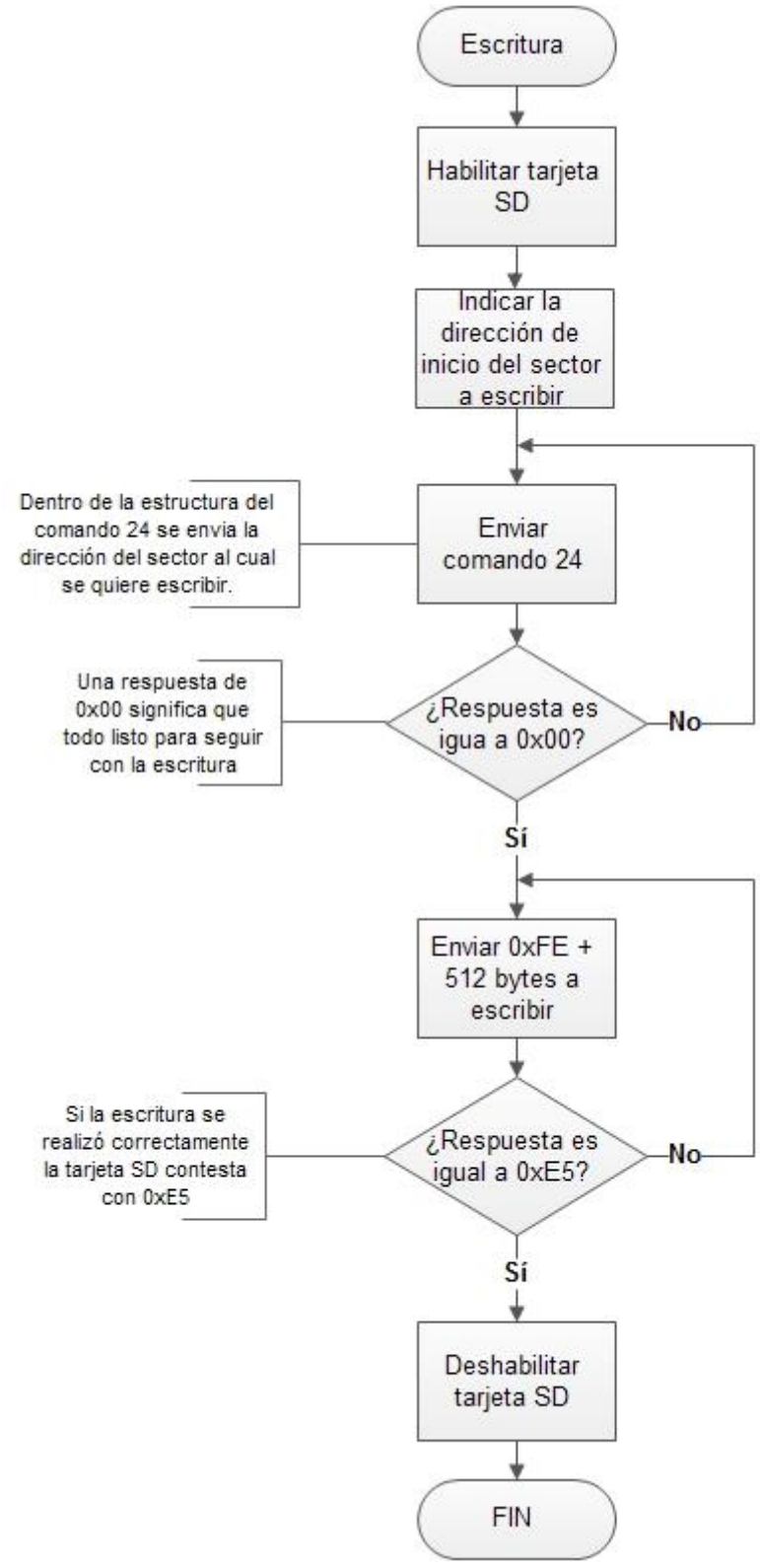

Figura 2.14: Proceso de escritura en la tarjeta SD.
Fuente [Autor]

Los métodos para la manipulación de la tarjeta SD mostrados, hasta este punto, la utilizan como cualquier circuito integrado de almacenamiento persistente y en general sería el más rápido, pero se presenta el problema que la dirección de inicio del sector o sectores para el almacenamiento de datos puede diferir de una tarjeta a otra (aunque sean del mismo fabricante), lo que generaría la imposibilidad de la lectura o escritura en la misma, para evitar esta incertidumbre se debe aprovechar el sistema de formateo que posee la memoria SD, en concreto FAT 16, que nos proporciona información que facilita la manipulación de dicha tarjeta SD.

Para la implementación del sistema FAT 16 se utiliza una librería desarrollada por un aficionado que responde al seudónimo de "Suky", que cuenta con una serie de funciones para el manejo de la tarjeta SD. A continuación se hace una breve descripción del uso de las funciones más importantes que se deben conocer.

Funciones disponibles y sus parámetros:

Buscando una dirección de carpetas.

Por ejemplo si en nuestra memoria SD tenemos la siguiente estructura de directorios "/Logger/temp/data", y deseamos crear una carpeta o archivo en ella debemos determinar el clúster donde se encuentra definida para trabajar en ella. Para ello utilizamos la siguiente función:

FAT_FindDirectory (Direccion, DirectorioInicial, UbicacionCarpeta);

Dónde:

Dirección: indica la estructura de directorios a buscar ("/LOGGER/TEMP/DATA").

DirectorioInicial: Indica a partir de que carpeta buscamos estructura. En principio iniciaríamos en la carpeta raíz y este valor seria 0.

UbicacionCarpeta: En esta variable se devuelve el clúster donde trabajar.

NOTA: La búsqueda se realiza a partir de los nombres cortos.

Crear una carpeta.

Para ello primero definimos el nombre corto (Mayúscula) y el nombre largo, que puede ser el mismo.

NombreCorto[12]="CARPETA"

NombreLargo[30]="Carpeta creada con PIC"

Utilizamos la siguiente función:

FAT_CreateDirectory (NombreLargo, NombreCorto, Ubicación);

Dónde:

Ubicación: es la dirección del Clúster en donde la crearemos. Esta dirección puede ser DirectorioRaiz (Definida dentro de la librería como 0) o una que se obtiene a buscar una dirección de carpetas con FAT_FindDirectory (…).

Retorna: La dirección donde se ha creado la carpeta para posterior trabajo dentro de ella (Crear un archivo, otra carpeta, etc.…)

Creación de un archivo.

Es idéntico a crear una carpeta pero con el adicional de indicar el string a escribir en dicho archivo. Dadas las capacidades físicas del microcontrolador el tamaño del string siempre será menor a un clúster.

FAT_CreateFile (NombreLargo, NombreCorto, Ubicacion, Texto);

Agregar datos a un archivo.

Si tenemos ya un archivo creado con datos almacenados y necesitamos agregar más datos con esta función es relativamente sencillo donde podemos ir generando un archivo de cualquier tamaño. La función a utilizar es la siguiente:

FAT_OpenAddFile (NombreArchivo, UbicacionCarpeta, Texto);

Dónde:

NombreArchivo: Nombre corto de archivo en donde agregaremos datos.

UbicacionCarpeta: Dirección de carpeta contenedora del archivo.

Texto: Texto a agregar al archivo.

Abrir un archivo y leer su contenido:

En la librería se lee el archivo y se envía por puerto serial, es aquí donde se debe cambiar a gusto.

FAT_OpenFile (NombreArchivo, UbicacionCarpeta);

NombreArchivo: Nombre corto de archivo en donde leeremos datos.

UbicacionCarpeta: Dirección de carpeta contenedora del archivo.

Ejemplo de uso de la librería (compilador CCS):

```
#include <18F4620.h>
#device adc=8
#fuses H4,NOWDT,NOPROTECT,NOLVP,NODEBUG
#use delay(clock=40000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

#include "FAT16.c"

UINT8 Texto[512]="Texto Creado con PIC....\r\n";

void main(){
   int1 Card_Work=0;
   char NombreCorto[13];
   char NombreLargo[50];
   UINT16 UbicacionFolder=0;
   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF);
   setup_psp(PSP_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED|T1_DIV_BY_1);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   InitHard_SDCard();
   delay_ms(1000);
   while(1){
      if((SD_DETEC==0)&&(Card_Work==FALSE)){
```

```
         delay_ms(500);
         if(SD_DETEC==0){
            Card_work=1;
            SDCard_init();
            FAT_init();
            strcpy(&NombreCorto[0],"CARPET~1");
            strcpy(&NombreLargo[0],"Carpeta de PIC");
UbicacionFolder=FAT_CreateDirectory(&NombreLargo[0],&NombreCorto[0],DirectorioRaiz);
            strcpy(&NombreCorto[0],"ARCHIV~1.txt");
            strcpy(&NombreLargo[0],"Archivo con PIC.txt");

       FAT_CreateFile(&NombreLargo[0],&NombreCorto[0],UbicacionFolder,&Texto[0]);
                     FAT_OpenAddFile(&NombreCorto[0],UbicacionFolder,&Texto[0]);
                     }
               }

      if((SD_DETEC==1)&&(Card_Work==TRUE)){
         Card_work=0;
               }
         }
}
```

2.7. RELOJ DE TIEMPO REAL.

Se dota a la placa de medición con un reloj de tiempo real a base de un circuito integrado DS1307 de la empresa Maxim, con el propósito de que no esté atada al equipo concentrador y se pueda utilizar para medición independiente. En la sección 1.8 se explica el funcionamiento de los circuitos RTC y la hoja técnica de este circuito integrado se encuentra en los anexos.

Se emplea este circuito integrado porque es de los pocos disponible en el país que realizan la función de reloj calendario. El circuito electrónico necesario para el empleo de este circuito integrado se muestra en la figura 2.15.

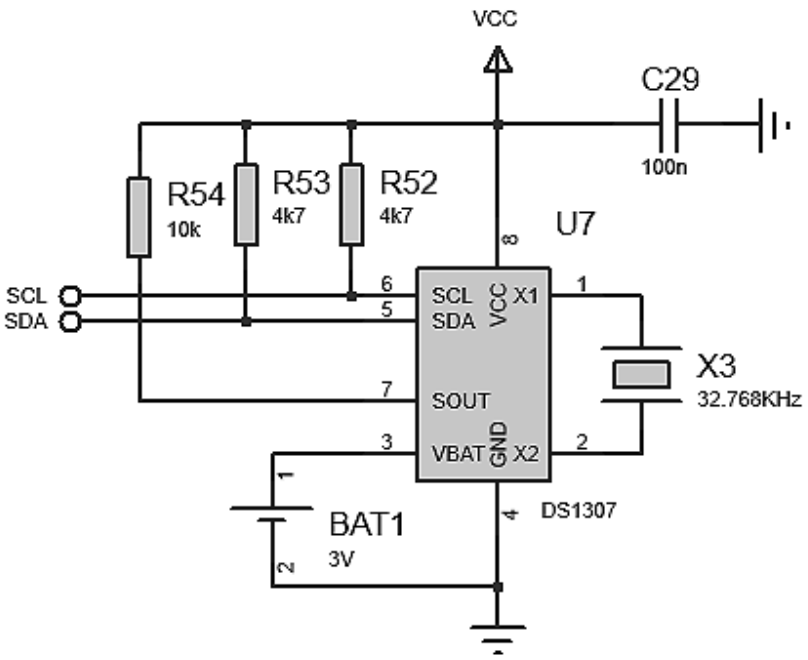

Figura 2.15: Circuito electrónico del reloj de tiempo real.
Fuente [Autor]

2.7.1. FIRMWARE PARA EL MANEJO DEL MÓDULO RTC.

La rutina empleada para iniciar el módulo RTC y establecer la fecha y hora se muestra en la figura 2.16. Para fijar la fecha y la hora solo se debe enviar la dirección del registro de inicio y cada escritura posterior corresponde al próximo registro, tal cual funciona el bus I2C.

Los diagramas de flujo de la figura 2.17 corresponden a las rutinas para lectura de la fecha y la hora actual.

Es importante tener presente que los datos que maneja el circuito integrado DS1307 están en formato decimal codificado en binario (BCD por sus siglas en inglés).

El archivo ds1307.c contiene todas las funciones necesarias para el manejo del circuito RTC, el cual puede apreciar en los anexos.

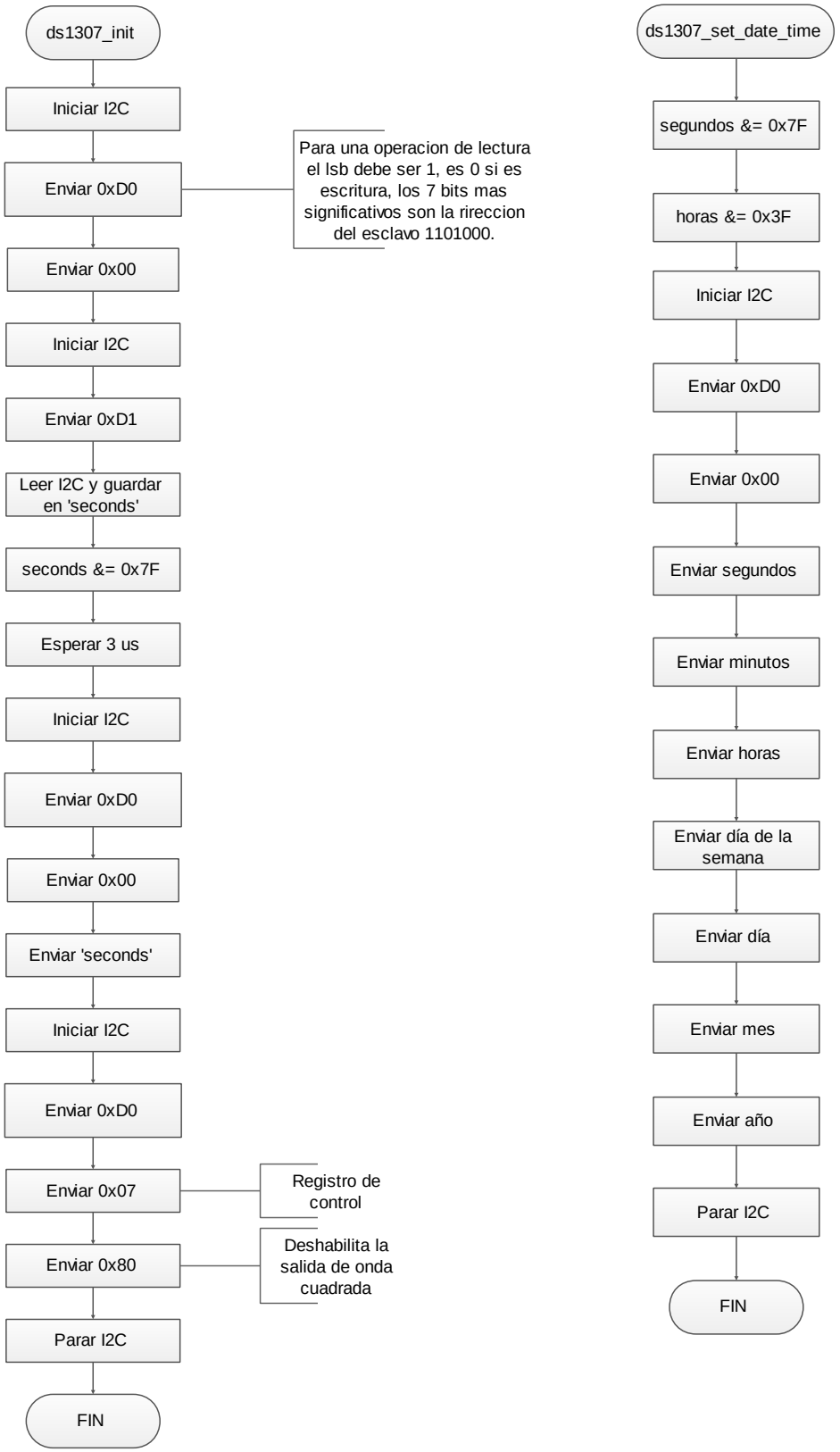

Figura 2.16: Rutina de inicialización y la fijación de la fecha y hora del DS1307. Fuente [Autor]

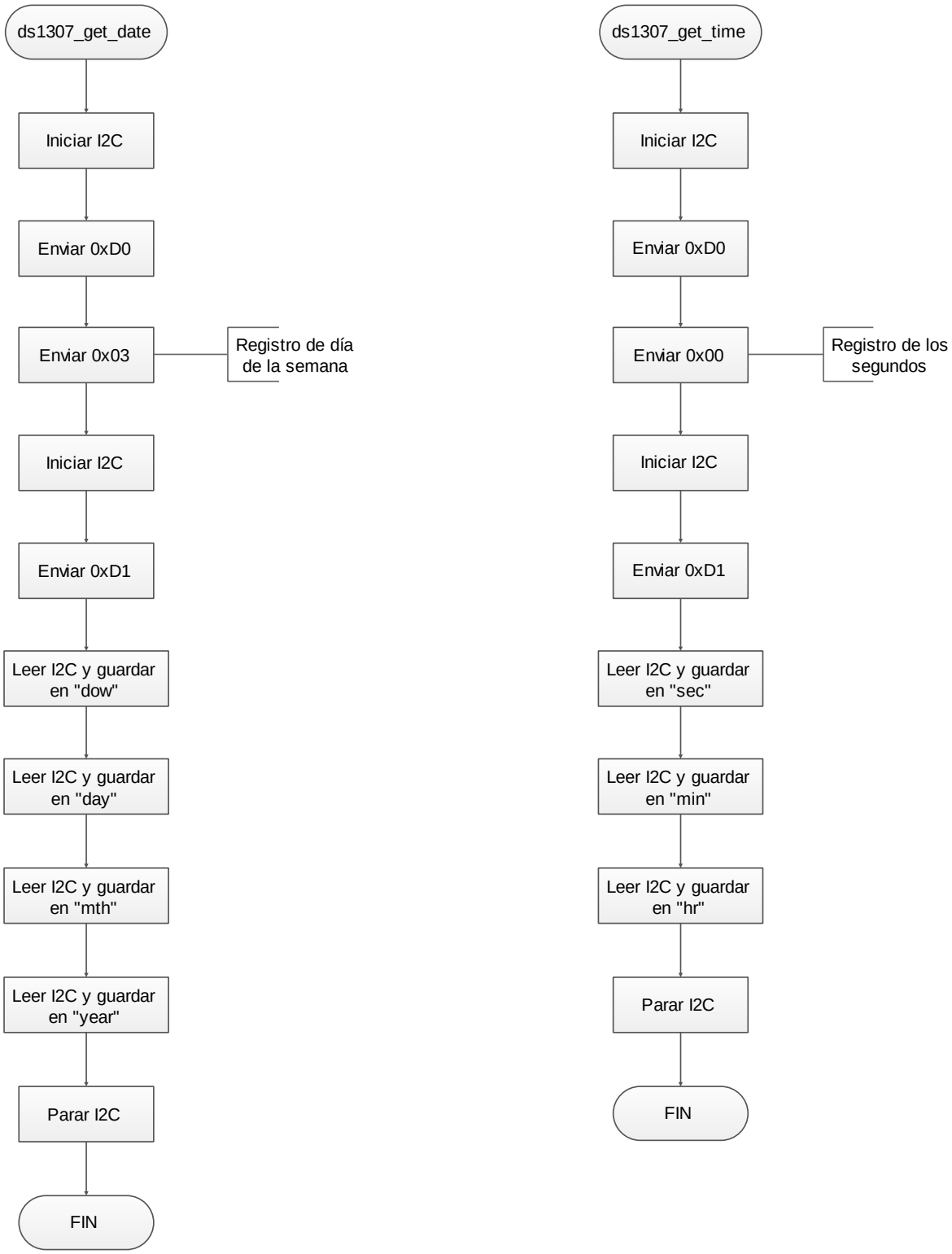

Figura 2.17: Obtener fecha y hora actual en el DS1307.
Fuente [Autor]

2.8. VISUALIZACIÓN DE DATOS.

Se emplea una pantalla de cristal líquido (LCD) de 2 líneas y 16 caracteres, en esta se muestran las mediciones instantáneas de la corriente y voltaje rms, también se emplea para notificar errores que se presenten en el sistema implementado.

La ventaja que presenta la pantalla de cristal líquido es el bajo consumo de potencia, fácil conexión con el microcontrolador y buena capacidad de visualización ya que con 2 líneas es posible observar los valores de voltaje y corriente por fase.

La comunicación con el microcontrolador se realizó mediante la transferencia de datos a través de sus líneas de datos y de control. Pueden usarse 4 u 8 líneas de datos. Si se escoge 8 líneas se facilita la programación en el microcontrolador, pero toca disponer de más puertos I/O del MCU. Para evitar el uso innecesario de pines I/O, se hace uso de un albur, el cual consiste en el empleo de un registro de desplazamiento, lo que permite controlar la pantalla LCD con tan solo 3 pines I/O, ver figura 2.7.

Este artilugio conlleva a que se tenga una mayor latencia en la visualización de los datos y aunque para esta aplicación no resulta en un problema, para otro tipo de aplicación donde se necesita una mayor rapidez en la visualización sí representa un inconveniente.

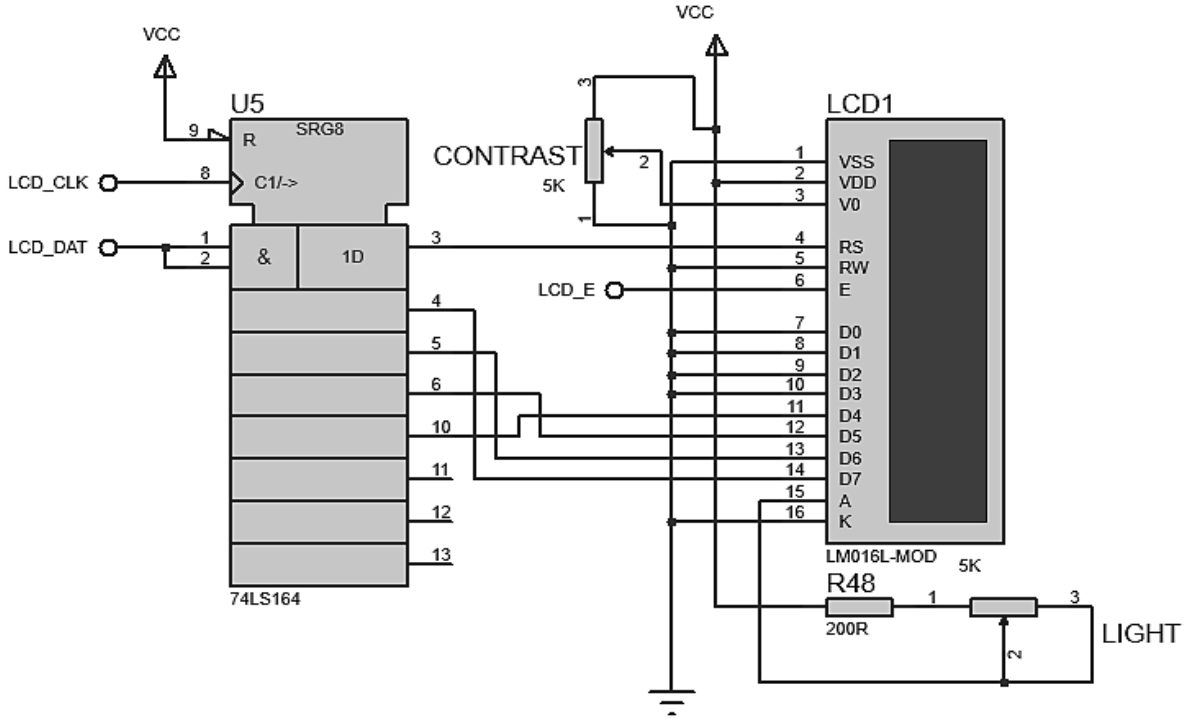

Figura 2.18: Circuito de control de la pantalla LCD con 3 pines I/O.
Fuente [Autor]

2.8.1. FIRMWARE PARA EL MANEJO DE LA PANTALLA LCD.

Se emplea la librería que modifico AKENAFAB Y Duende_azul del foro todopic [2].

Es una modificación a la librería flex_lcd, la lcd se programa en modo de 4 bits, para lograr esto se utiliza un Registro de Corrimiento: 74LS164.

Los pines utilizados son:

- **LCD_E**: Señal de control Enable del LCD
- **LCD_CK**: Señal de reloj del registro de corrimiento.
- **LCD_DAT**: Salida del bit menos significativo hacia el registro de desplazamiento

Estos pines necesitan ser definidos antes de llamar a la librería:

#define LCD_E PIN_A0 // Cambiar A0, A1 y A2
#define LCD_CK PIN_A1 // por los pines
#define LCD_DAT PIN_A2 // que gusten

Las funciones más importantes de esta librería se muestran a continuación:

```
void lcd_send_nibble(int8 nibble, int rs_bit)
{

int x;
if(RS_bit==1)
   nibble=nibble|0x10;

for(x=0;x<5;x++){
         output_bit(LCD_DAT,shift_right(&nibble,1,0));
         delay_cycles(1);
         output_low(LCD_CK);
         delay_us(1);
         output_high(LCD_CK);}

 output_high(LCD_E);
 delay_us(2);
 output_low(LCD_E);
}

//-----------------------------------------
// Send a byte to the LCD.
```

```
void lcd_send_byte(int8 address, int8 n)
{
//output_low(LCD_RS);
RS_bit=0;
delay_us(100);


if(address)
   //output_high(LCD_RS);
   RS_bit=1;
else
   //output_low(LCD_RS);
   RS_bit=0;

 delay_cycles(1);


output_low(LCD_E);

lcd_send_nibble(n >> 4,RS_bit);
lcd_send_nibble(n & 0xf,RS_bit);
}

//----------------------------
void lcd_init(void)
{
int8 i;

//output_low(LCD_RS);
RS_bit=0;

output_low(LCD_E);

delay_ms(20);

for(i=0 ;i < 3; i++)
   {
    lcd_send_nibble(0x03,RS_bit);
    delay_ms(5);
   }

lcd_send_nibble(0x02,RS_bit);

for(i=0; i < sizeof(LCD_INIT_STRING); i++)
   {
    lcd_send_byte(0, LCD_INIT_STRING[i]);

    delay_ms(5);

   }

}
//----------------------------

void lcd_gotoxy(int8 x, int8 y)
{
int8 address;

if(y != 1)
   address = lcd_line_two;
else
   address=0;
```

```
address += x-1;
lcd_send_byte(0, 0x80 | address);
}
//-----------------------------
void lcd_putc(char c)
{
 switch(c)
   {
    case '\f':                //limpia pantalla
      lcd_send_byte(0,1);
      delay_ms(8);
      break;

    case '\n':                //cambio de linea
       lcd_gotoxy(1,2);
       break;

    case '\b':                //retrocede 1 caracter
       lcd_send_byte(0,0x10);
       break;

    default:
       lcd_send_byte(1,c);
       break;
   }
}
```

2.9. MÓDULOS DE COMUNICACIÓN.

Se necesita crear un medio para la intercomunicación de todos los módulos que componen el sistema implementado, en concreto se debe crear una forma para realizar configuraciones y la conexión con el modulo principal del proyecto SIPROE.

2.9.1. COMUNICACIÓN SERIAL RS-232.

Para modificar el reloj calendario, notificación más explícita de errores y/o depuración del sistema, se hace uso del módulo EUSART[14] para la comunicación por puerto serie que dispone el microcontrolador.

La comunicación por puerto serie se realiza utilizando el estándar recomendado 232 (RS-232), esto origina la necesidad del uso de un transceptor que realiza la

[14] EUSART: Enhanced Universal Synchronous Asynchronous Receiver Transmitter por sus siglas en inglés, transmisor receptor síncrono asíncrono universal mejorado.

función de convertir los niveles de voltaje TTL a los niveles que establece el estándar que van desde los -12Volts a los +12Volts, esto se puede hacer con un circuito transistorizado o con un circuito integrado especializado, de los cuales hay varias opciones en el mercado, como el MAX232 de la empresa Maxim utilizado para este módulo, ver figura 2.19.

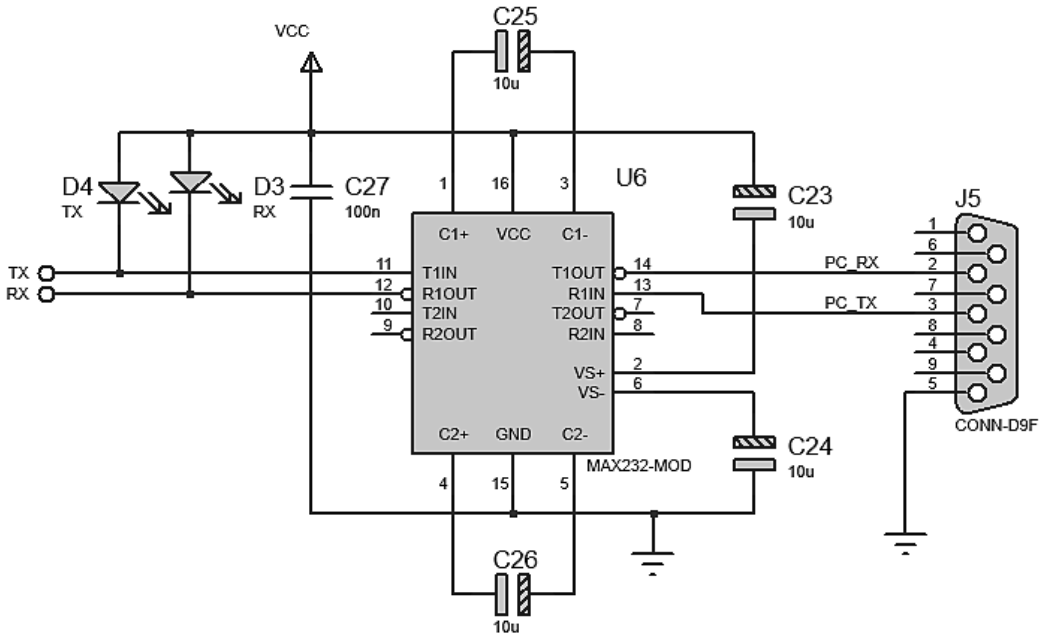

Figura 2.19: Modulo de comunicación RS-232.
Fuente [Autor]

Para implementar el protocolo RS-232 se usan las librerías del compilador de CCS, cuyo uso se explican a continuación.

#USE RS232 (BAUD=baudios, XMIT=pin, RCV=pin...)

Esta directiva le dice al compilador la velocidad en baudios y los pines utilizados para la I/O serie. Esta directiva tiene efecto hasta que se encuentra otra directiva RS232.

La directiva **#USE DELAY** debe aparecer antes de utilizar **#USE RS232**. Esta directiva habilita el uso de funciones tales como GETCH, PUTCHAR y PRINTF. Si la

I/O no es estándar es preciso poner las directivas **FIXED_IO** o **FAST_IO** delante de **#USE RS232**.

En la tabla 2.4 se muestran las diferentes opciones que presenta el compilador con las que se puede configurar el módulo RS-232.

Tabla 2.4: Opciones de configuración del módulo RS232.
Fuente [21]

RESTART_WDT	Hace que GETC () ponga a cero el WDT mientras espera un carácter.
INVERT	Invierte la polaridad de los pines serie (normalmente no es necesario con el convertidor de nivel, como el MAX232). No puede usarse con el SCI[15] interno.
PARITY=X	Donde X es N, E, u O.
BITS=X	Donde X es 5-9 (no puede usarse 5-7 con el SCI).
FLOAT_HIGH	Se utiliza para las salidas de colector abierto.
ERRORS	Indica al compilador que guarde los errores recibidos en la variable RS232_ERRORS para restablecerlos cuando se producen.
BRGH1OK	Permite velocidades de transmisión bajas en chips (uC's, memorias, etc) que tienen problemas de transmisión. Cuando utilizamos dispositivos con SCI y se especifican los pines SCI, entonces se usará el SCI. Si no se puede alcanzar una tasa de baudios dentro del 3% del valor deseado utilizando la frecuencia de reloj actual, se generará un error.
ENABLE=pin	El pin especificado estará a nivel alto durante la transmisión.
FORCE_SW	Usa una UART software en lugar del hardware aun cuando se especifican los pines del hardware. La definición de RS232_ERRORS es como sigue: **Sin UART:** El bit 7 es el 9º bit para el modo de datos de 9 bit. El bit 6 a nivel alto indica un fallo en el modo flotante alto. **Con UART:** Usado sólo para conseguir: Copia del registro RCSTA, excepto: que el bit 0 se usa para indicar un error de paridad. **Ejemplo:** *#use rs232(baud=9600, xmit=PIN_A2,rcv=PIN_A3)*

[15] SCI es el acrónimo en inglés de Interfaz de comunicación Serie.

2.9.2. COMUNICACIÓN BUS CAN.

La comunicación del concentrador, a base del proyecto SIPROE, es por medio del protocolo bus CAN, el microcontrolador posee un módulo para el uso de este protocolo, pero al igual que en el caso del módulo RS-232, se necesita el uso de un transceptor que cumple con la función de la capa física del modelo OSI visto en el capítulo 1. Según el protocolo se necesita emplear un cable trenzado para la conexión y para ello se usa un cable UTP, dicho esto se agrega un conector RJ-45 en este módulo, cuyo diagrama del circuito electrónico se muestra en la figura 2.20.

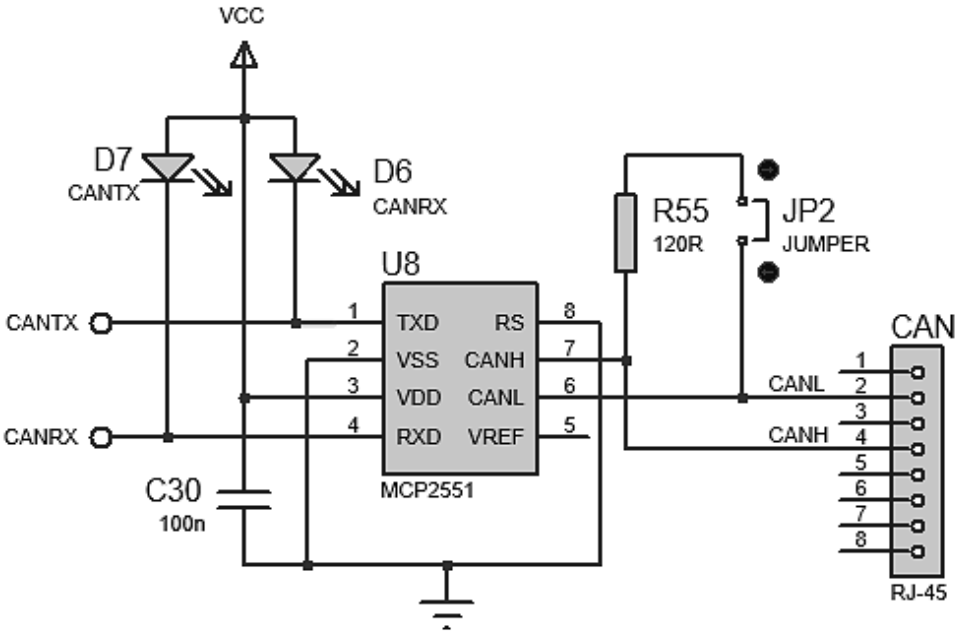

Figura 2.20: Modulo de comunicación bus CAN.
Fuente [Autor]

Para el manejo del bus CAN es necesario incluir un grupo de archivos de librería proporcionadas por Microchip, estos archivos se han modificado para sean utilizados por el compilador C de CCS.

La tabla 2.5 muestra una breve descripción de la función de los archivos originales usados.

Tabla 2.5: Archivos necesarios para la comunicación modbus sobre el bus CAN. Fuente [Autor]

Archivo	Descripción
Can18xx8.asm	Librería CAN proporcionada por MICROCHIP
funcionesModbusM0.asm	Librería modbus implementa funciones 02, 03, 05 y 16
CAN18xx8.inc	Lista de funciones CAN que pueden ser utilizar, solo es necesario incluir este archivo en el archivo .asm que se utiliza
CANDef.inc	Archivo donde se realizan las definiciones de variables, reserva de memoria y definición de macros utilizadas por la librería CAN
Definiciones1.inc	Archivo donde se definen métodos, macros y variables utilizadas por las librerías I2C, time y funcionesModbusM0
Modbuslink.inc	Lista de funciones modbus que pueden ser utilizadas, para utilizar las funciones es necesario que en la librería estén definidas como global.
Var_modbus.inc	Definición de variables utilizadas por la librería modbus
Variables.inc	Definición de variables generales

A continuación se presenta una sección de firmware para la configuración del CAN:

```
;*****************************************************************************
;               CONFIGURACION DEL PROTOCOLO CAN                                 *
;               configuracion inicial 125kbps@10MHz all valid messages          *
;               CANInitialize  SJW, BRP, PHSEG1, PHSEG2, PROPSEG, Flags *
;               PHSEG1+PHSEG2+PROPSEG=          #TQ-1                           *
;*****************************************************************************
Init_CAN:
                CLRWDT
                CANInitialize           1, 5, 3, 3, 1,CAN_CONFIG_ALL_VALID_MSG ; Inicializar el Modulo a 125kbps a
10MHz
                ;CANInitialize          1, 1, 3, 3, 1,CAN_CONFIG_ALL_VALID_MSG ; configuracion con 4Mhz, 125kbps
#ifdef  RX_TEST
;Set configuration mode
                CANSetOperationMode             CAN_OP_MODE_CONFIG      ; Activar modo Configuracion
;configurar Mascara B0 to 0xffffeff=11111111111111111111011111111
                CANSetReg CAN_MASK_B0, 0xFFFFFEFF, CAN_CONFIG_XTD_MSG
                ;CANSetReg              CAN_MASK_B0, 0xFFFFFEFF, CAN_CONFIG_STD_MSG
;Set Filter 0 with
                CANSetReg CAN_FILTER_B0_F1, 0x011, CAN_CONFIG_XTD_MSG       ;mensajes del CONTROLADOR
DE ACC
;Set Filter 1 with
                CANSetReg CAN_FILTER_B0_F2, 0x22, CAN_CONFIG_XTD_MSG        ;mensajes del CONTROLADOR
A/C E ILUMINACION
        CANSetReg CAN_MASK_B1,0xFEFFFF7F, CAN_CONFIG_XTD_MSG
;Set Filer 2 with 0x33
        CANSetReg CAN_FILTER_B1_F1,0x33, CAN_CONFIG_XTD_MSG                 ;MENSAJES DEL RELOJ
```

Este segmento de código fuente realiza la configuración del protocolo CAN, con la función CANinitialize, la cual recibe 6 parámetros. Con estos se fija una comunicación de 125Kbps a 10Mhz. También se configuran los filtros CAN en los que escucha a los controladores, la función CANsetReg recibe como parámetro el filtro o la máscara del buffer a configurar, el ID del filtro y el tipo de filtro estándar o extendido, para nuestro caso extendido.

El controlador maestro escucha a los demás controladores en diferentes filtros por ejemplo: al control de acceso lo escucha en el filtro uno del buffer cero con ID=0x11, al controlador de propósito general lo escucha en el filtro dos del buffer cero con ID=0x22.

2.10. FUENTE DE ALIMENTACIÓN.

La alimentación a la placa del medidor se realiza por medio de una fuente de voltaje DC externa, solo se dota a la placa con una etapa de regulación de voltaje a la cual se accede por medio un conector (Jack DC), esto permite la conexión de una fuente de voltaje DC dentro del rango de 9 a 40 VDC, la regulación se realiza por medio de dos circuitos integrados especializados, el LM317.

Uno de los reguladores se ajusta para que en su salida se tengan 5.1 VDC en circuito abierto, lo que nos asegura que bajo carga se estará suministrando 5 VDC a los circuitos integrados en uso dentro de la placa del medidor.

La tarjeta SD funciona con una alimentación de 2.7 a 3.6 VDC, esto genera el uso de un segundo regulador de voltaje que se coloca contiguo al primer regulador y se ajusta a un voltaje de circuito abierto de 3.3 VDC.

En la figura 2.21 se aprecian otros dispositivos aparte de los susodichos reguladores de voltaje, que desempeñan la función de protección, el encendido y la visualización de la presencia de una fuente de alimentación.

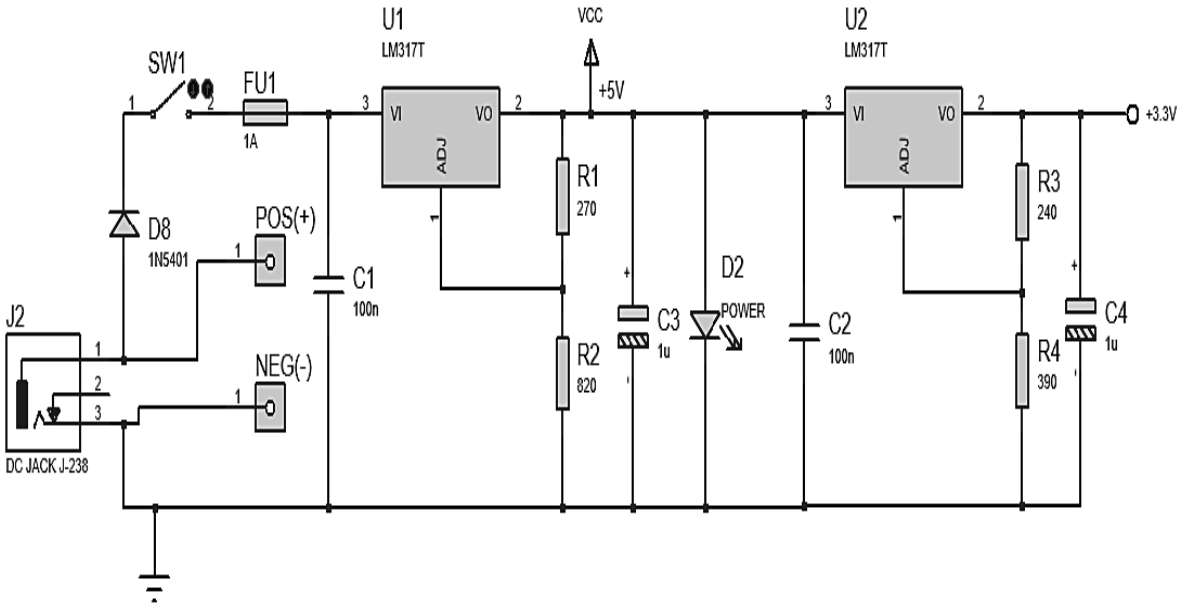

Figura 2.21: Regulación de la fuente de voltaje externa.
Fuente [Autor]

CAPITULO III.

DISEÑO Y DESARROLLO DEL SITIO WEB

Introducción.

En este capítulo se describe detalladamente las partes involucradas en el diseño y la implementación del Sitio Web tomando criterios de diseño, basados en tecnología JAVA.

La aplicación web, es la que permite a un usuario a través de una interfaz gráfica amigable, administrar, controlar y monitorear sistemas de iluminación, aire acondicionado y control de acceso que son las diferentes opciones que el sistema SIPROE proporciona. Para con el fin de ahorrar energía eléctrica.

3.1. FUNDAMENTOS DE DISEÑO: ENFOQUE SOFTWARE.

Retomando la funcionalidad de su versión predecesora, el módulo electrónico SACME V1.0, el prototipo SIPROE V1.0 cuenta con una arquitectura cliente servidor, la cual a través de un aplicativo WEB desarrollado en ambiente JAVA y con funcionalidad de cliente, denominado CLIENTE SIPROE, ejecuta peticiones hacia y recibe respuestas desde un controlador maestro denominado CONTROLADOR SIPROE M0, el cual lleva a cabo las siguientes funcionalidades:

- Sirve como GATEWAY o puente de enlace entre el CLIENTE SIPROE y los controladores de funcionalidad, los cuales se describirán más adelante.

- Ejecuta la funcionalidad de servidor MODBUS, recibiendo peticiones MODBUS, las cuales al ser recibidas por el CONTROLADOR SIPROE M0, son transmitidas tanto al módulo SACME V1.1 como a controladores de funcionalidad para ser ejecutadas por estos.
- Una vez ejecutada la petición MODBUS, por el módulo SACME V1.1 o los controladores de funcionalidad, el CONTROLADOR SIPROE M0 transmite las respuestas MODBUS hacia el cliente.
- Aunque la arquitectura maestro esclavo no es aplicable a una red CAN, al haberse desarrollado el protocolo MODBUS sobre CAN, el CONTROLADOR SIPROE M0, es en esencia un controlador MAESTRO dentro de la arquitectura de SIPROE V1.0; ya que se encarga de re direccionar las funcionalidades implementadas de MODBUS hacia los controladores de funcionalidad y recibir de estos la respuesta a la petición del CLIENTE SIPROE.
- Ejecuta las funcionalidades de administración de la red CAN.

El prototipo SIPROE está diseñado con tecnología JAVA para entornos empresariales, en tal sentido cuenta con una arquitectura de diseño multicapas, definida como se muestra en la Figura 3.1

Adicionalmente SIPROE incorpora parte del hardware de SACME tal como el Gateway MODBUS TCP/RTU para la comunicación con el cliente del sistema (usuario); El controlador maestro incorpora COMPONENTES de SACME para el manejo completo de las funcionalidades de SACME y un transceptor CAN para la comunicación con los demás controladores de funcionalidad; cada controlador de funcionalidad adicionalmente incorpora un transceptor individual que lo conecta al bus CAN y le permite comunicarse a través de ese medio físico a los demás controladores. Se observa en la Figura 3.1 que la interface I2C proporciona a cada controlador de funcionalidad la posibilidad de comunicarse con el entorno físico para el cual está diseñado.

3.1.1. ASPECTOS GENERALES EN ARQUITECTURA WEB.

A la hora de abordar el desarrollo del sitio web, hay una serie de consideraciones acerca del mismo que hay que tener muy presentes, dado que son claves en el tipo de diseño y metodologías de desarrollo a aplicar.

3.1.2. ESCALABILIDAD.

Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. En general, también se podría definir como la capacidad del sistema de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes. Por ejemplo la implementación de una nueva funcionalidad del sistema o la incorporación de más hardware, o también los servicios pueden separarse en distintos puntos en la red.

3.1.3. SEPARACIÓN DE RESPONSABILIDADES.

Se logra a través de la separación en capas del sistema. Distintas responsabilidades no deben ser delegadas en la misma clase, y llevado esto algo más allá, en el mismo conjunto de clases. En la actualidad, la tendencia más aceptada es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas que interaccionan unas con otras a través de sus interfaces. Se trata de los sistemas denominados multicapa o *n*-capas. Aplicados a los proyectos web, el modelo más básico es el de aplicaciones 3 capas: *presentación, negocio o dominio y capa de acceso a datos.*

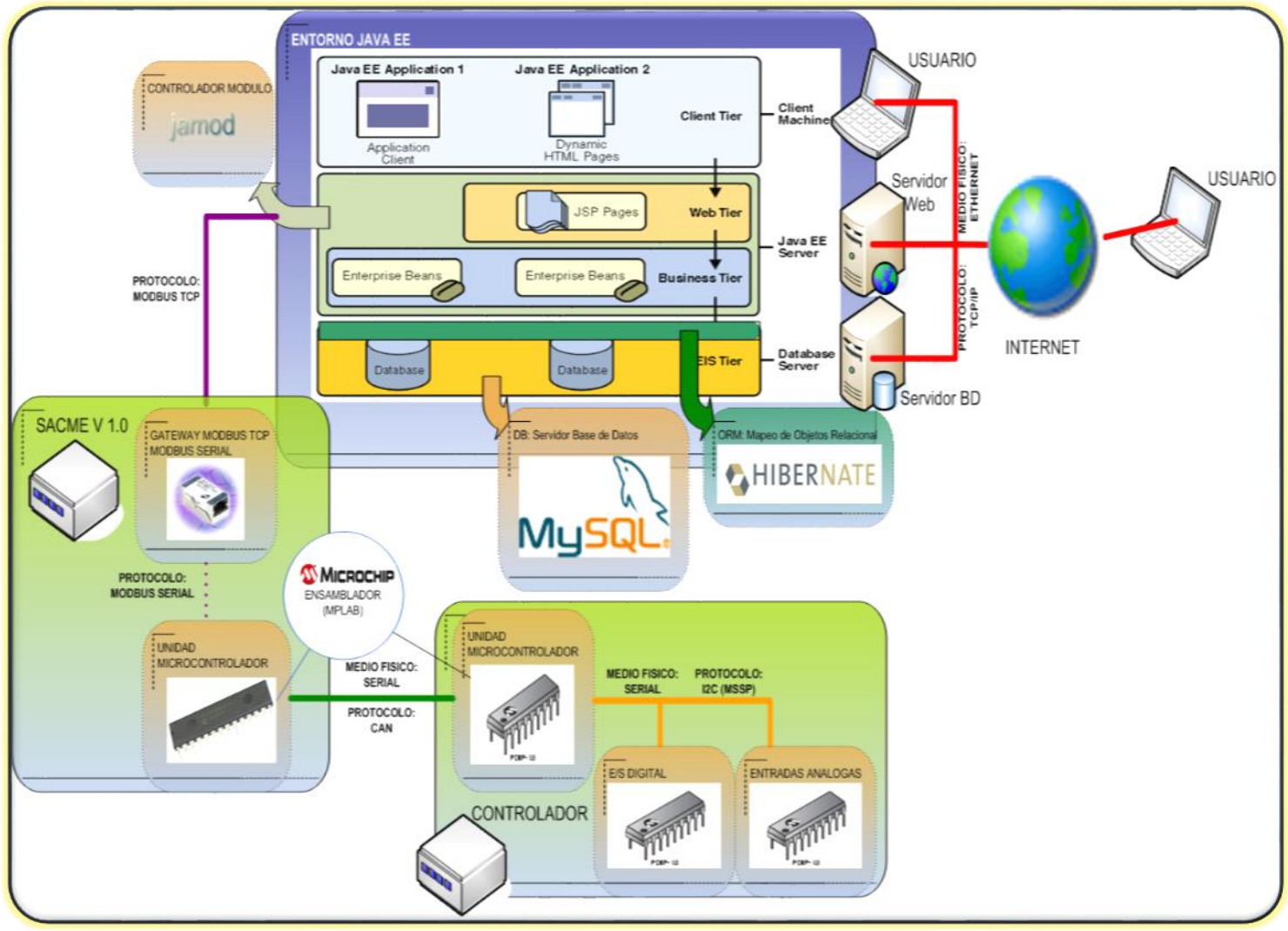

Figura 3.1: Entorno de la aplicación visto desde la perspectiva Software.
Fuente [10]

3.1.4. PORTABILIDAD.

En la medida de lo posible, la aplicación web debe poder adaptarse a las distintas posibles arquitecturas físicas susceptibles de ser empleadas para el despliegue del paquete, limitándose en la medida de lo posible el impacto de tal adaptación a tareas de configuración, y evitándose así la necesidad de modificar el código de la misma ante dichas situaciones.

3.1.5. GESTIÓN DE LA SESIÓN DEL USUARIO, CACHEADO DE ENTIDADES.

Con objeto de limitar en la medida de lo posible los accesos innecesarios a memoria secundaria (bases de datos, ficheros externos de configuración, etc.), se propone un sistema que se apoya en parte en el empleo de la sesión HTTP(s) para cachear ciertos datos referentes a la sesión del usuario, o bien comunes a todas las sesiones de usuario. Obviamente, la cantidad y naturaleza de las entidades susceptibles de ser cacheadas será determinada teniendo muy presentes aspectos de rendimiento del producto, dado que un empleo no apropiado de esta técnica puede y suele llevar a un consumo excesivo de los recursos del sistema (memoria).

3.1.6. APLICACIÓN DE PATRONES DE DISEÑO.

El empleo y aplicación de patrones de diseño facilita el entendimiento del código y, por tanto, reduce considerablemente el coste de mantenimiento, dado que además de aportar soluciones eficientes para problemas comunes, son muy interesantes como medio de entendimiento entre diseñadores e implementadores.

3.1.7. SEPARACIÓN LÓGICA EN CAPAS.

A la hora de plantear el diseño de la aplicación web, el primer paso es conseguir separar conceptualmente las tareas que el sistema debe desempeñar entre las distintas capas lógicas y en base a la naturaleza de tales tareas se ha de partir de la separación

inicial en tres capas, diferenciando que proceso de los que hay que modelar responde a tareas de presentación, cual a negocio y cual a acceso a datos. En caso de identificar algún proceso lógico que abarque responsabilidades adjudicadas a dos o más capas distintas, es probable que dicho proceso deba ser explotado en subprocesos iterativamente, hasta alcanzar el punto en el que no exista ninguno que abarque más de una capa lógica.

3.1.8. CAPA DE PRESENTACIÓN.

Es la responsable de todos los aspectos relacionados con la interfaz de usuario de la aplicación. Así, en esta capa de resuelven cuestiones como:

- Navegabilidad del sistema, mapa de navegación, etc.
- Formateo de los datos de salida: Resolución del formato más adecuado para la presentación de resultados. Está relacionado directamente con la internacionalización de la aplicación.
- Internacionalización: Los textos, etiquetas, y datos en general a presentar se obtendrán de uno u otro fichero de recursos en base al idioma preferido del navegador del usuario. En base a esta condición se ven afectadas las representaciones numéricas, las validaciones sobre los datos de entrada (coma decimal o punto decimal) y otros aspectos relativos al idioma del usuario remoto.
- Validación de los datos de entrada, en cuanto a formatos, longitudes máximas, etc.
- Interfaz gráfica con el usuario.
- Multicanalidad de la aplicación: Una misma aplicación web puede contar con varias presentaciones distintas, determinándose el uso de la adecuada en base al dispositivo visualizador desde el que trabaje el usuario. Así, no se representará la misma información con el mismo formato en un

Navegador web estándar que en un dispositivo móvil provisto de un navegador WAP[16].

3.1.9. CAPA DE NEGOCIO

En esta capa es donde se deben implementar todas aquellas reglas obtenidas a partir del análisis funcional del proyecto. Así mismo, debe ser completamente independiente de cualquiera de los aspectos relacionados con la presentación de la misma. De esta forma, la misma capa de negocio debe poder ser empleada para una aplicación web común, una aplicación WAP, o una StandAlone. Por otro lado, la capa de negocio ha de ser también completamente independiente de los mecanismos de persistencia empleados en la capa de acceso a datos. Cuando la capa de negocio requiera recuperar o persistir entidades o cualquier conjunto de información, lo hará siempre apoyándose en los servicios que ofrezca la capa de acceso a datos para ello. De esta forma, la sustitución del motor de persistencia no afecta lo más mínimo a esta parte del sistema. Debería poder reemplazarse el gestor de bases de datos por un conjunto de ficheros de texto sin necesitar tomar ni una línea de código de presentación o negocio.

3.1.10. IMPLEMENTACIÓN DE LOS PROCESOS DE NEGOCIO IDENTIFICADOS EN EL ANÁLISIS DEL PROYECTO.

Como se deduce del párrafo anterior, los procesos de negocio implementados en esta capa son totalmente independientes de cualquier aspecto relativo a la presentación de los mismos. Pongamos por ejemplo la generación de un informe que conste de varias filas las cuales deberán sombrearse con un color determinado por la

---

[16] **Wireless Application Protocol** o **WAP** (protocolo de aplicaciones inalámbricas) es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

superación o no de ciertos umbrales para ciertos valores. La aplicación de ciertos umbrales, y la consecuente determinación de la situación (correcta, incorrecta, etc.) de cada valor de la fila es un cálculo de negocio que debe afrontarse en esta capa. Sin embargo, sería en un error completar un campo adicional en el que, como resultado del cálculo, se determinara directamente el color de la fila. Lo adecuado es codificar la situación de la misma y, una vez en presentación, determinar el color adecuado en base al código recibido. De esta forma, si el usuario requiere que a partir de cierta fecha, el color rojo sea sustituido por el naranja, la modificación de este aspecto de presentación de información se limitaría en la aplicación a una modificación de la capa de presentación.

3.1.11. CONTROL DE ACCESO A LOS SERVICIOS DE NEGOCIO.

Dado que una misma aplicación puede contar con más de una capa de presentación al mismo tiempo, es aconsejable que la responsable última de ejecutar tareas sobre el control de acceso a los servicios del sistema no sea la capa de presentación, sino la de negocio. Los implementadores de la capa de negocio ni pueden ni deben confiar en que las futuras implementaciones de nuevas capas de presentación gestionen adecuadamente el acceso a los servicios de negocio. De esta forma, en cada invocación a cualquier método restringido de negocio se deberá comprobar por medio del sistema de autentificación adecuado, los derechos del usuario actual a realizar tal operación.

3.1.12. PUBLICACIÓN DE SERVICIOS DE NEGOCIO.

El lugar adecuado para que dos microaplicaciones o aplicaciones completas interactúen es a nivel de la capa de negocio. Así mismo, el modelo de colaboración recomendado en esta definición de arquitectura es el que se basa en el empleo de servicios web. De esta forma, la capa de negocio ofrecerá dos vistas alternativas, dado

que por un lado el conjunto de facades[17] con presentación ofrecerá a esta los servicios que se requieran para el funcionamiento de la microaplicación en sí, y por otro, conjunto de servicios serán publicados por medio (recomendablemente) de servicios web. Estos últimos estarán orientados a la colaboración con otros sistemas distintos u otras microaplicaciones pertenecientes al mismo proyecto, y deberán por tanto llevar un control más férreo sobre el acceso al servicio, dado que ahora, además de hacerlo a nivel de usuario, puede que sea necesario hacerlo a nivel de aplicación, por lo que cada aplicación remota debería estar explícitamente autorizada a invocar el servicio de negocio publicado.

3.1.13. INVOCACIÓN A LA CAPA DE PERSISTENCIA.

Los procesos de negocio son los que determinan que, como y cuando se debe persistir en el repositorio de información. Los servicios ofertados por la interfaz de la capa de acceso a datos son invocados desde la capa de negocio en base a los requerimientos de los procesos en ella implementados.

3.1.14. CAPA DE ACCESO A DATOS.

La capa de acceso a datos es la responsable de la gestión de la persistencia de la información manejada en las capas superiores. En esta capa, se definen las entidades o tablas para almacenar los datos de forma relacional, En un modelo académicamente purista, la interfaz de esta capa estaría compuesta por vistas de las entidades a persistir, pero a efectos prácticos, y con objeto de aprovechar la habitual potencia de los gestores de bases de datos, la interfaz muestra una serie de servicios que pueden agrupar operaciones en lo que se puede denominar "lógica de persistencia", como insertar usuario o inserción de roles, en la que podrían darse de alta al mismo tiempo

[17] El patrón de diseño *Facade* sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

un Rol y todos las entidades que dependan de dicho rol (porque no, el mismo usuario).

3.1.15. CAPA DE INFRAESTRUCTURA.

Esta capa, adyacente a todas las demás de la aplicación, comprende todos aquellos servicios susceptibles de ser requeridos desde cualquiera de las capas lógicas de la aplicación web. La gestión de un servicio y de las clases que lo implementan se realizará desde la concepción de un componente, es decir, una clase totalmente independiente de la aplicación que lo utiliza. La capa de infraestructura estará formada entonces por componentes, y por las clases gestoras necesarias para su configuración y gestión. De esta forma, cuando una clase de cualquiera de las capas lógicas de la aplicación requiera el uso de alguno de los servicios ofrecidos por la capa de infraestructura (por ejemplo, el servicio de Log), no tratará directamente con la clase que implemente tal servicio, sino que lo hará por medio del interfaz que cumpla la misma. Así mismo, la instanciación del servicio es responsabilidad de las clases gestoras de la capa de infraestructura. La clase cliente del servicio le pedirá a la capa de infraestructura que le facilite una instancia de la clase que implementa el servicio que necesita. La relación entre una interfaz que defina un servicio de infraestructura y la clase que implementa el servicio se establece en un fichero externo XML. De esta forma:

- La sustitución de un componente que implemente un servicio de la capa de infraestructura por otro distinto que cumpla la misma interfaz sólo requiere modificar el fichero de configuración.

- La configuración de cada uno de los componentes irá asimismo externalizada en ficheros XML.

- Se consigue desacoplar completamente la aplicación de su entorno de despliegue. En caso de que en un futuro tuviera que ser integrada con otros

sistemas, dicha tarea podría llegar a requerir sólo el desarrollo de los componentes adecuados o en encapsulamiento de los ya presentes en el sistema anfitrión. Volviendo al ya citado ejemplo del sistema de Log, es habitual que una compañía tenga normalizado el formato de salida del mismo para todos sus sistemas. Si se necesitará instalar el producto en otra compañía, que probablemente cuente también con su propio formato de trazas de Log, sólo se necesitaría encapsular las clases aportadas por la nueva compañía para la generación de trazas para adaptar su interfaz al que el servicio de la aplicación impone. Si además se tendiera al empleo de interfaces estándar (aunque esto no siempre es posible), esta tarea puede quedar reducida a una simple re-configuración del sistema.

- Las clases gestoras, en caso de que el comportamiento del componente lo permitiera, pueden trabajar con pools[18] de componentes para aquellos cuyo uso no implique un mantenimiento de estado, y sean susceptibles de invocarse con una frecuencia elevada.

- Las clases gestoras de la capa de infraestructura deben permitir establecer períodos de re-configuración, de forma que la alteración del comportamiento del sistema a través de sus componentes (sustitución y configuración de los mismos) se pueda hacer en caliente, evitando una parada en el sistema de producción. La modificación del comportamiento de ciertos componentes, como por ejemplo un pool de conexiones, facilita el sincronismo y dimensionamiento del sistema una vez entre en producción.

[18] En computación, se denomina **pool** (agrupamiento de conexiones) al manejo de una colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones.

Los servicios que deben según esta filosofía pertenecer a la capa de infraestructura son:

- Servicio de Log.
- Pool de conexiones JDBC[19] (o de cualquier otro sistema de persistencia).
- Sistema de configuración de la aplicación.
- Gestor de accesos/permisos de usuario a los distintos servicios de la aplicación.
- Otros más específicos del entorno del proyecto pero independientes del modelo.

3.2. IMPLEMENTACIÓN DEL SITIO WEB.

Para implementar el sitio web de Siproe que tiene las posibilidades de administrar, controlar y monitorear los recursos con funcionalidad de iluminación, aire acondicionado y acceso con el fin de ahorrar energía eléctrica en aplicaciones comerciales, una interfaz gráfica amigable al usuario es necesaria para interactuar con el hardware que en el capítulo anterior se desarrolló, aplicando las consideraciones del diseño del sitio web para ello, se detalla en esta sección el desarrollo de la aplicación web.

3.2.1. INDEPENDENCIA DE LA PLATAFORMA.

El lenguaje Java provee una máquina virtual o "procesador virtual" que ejecuta cualquier código que haya sido escrito en dicho lenguaje. Esto permite que el mismo binario ejecutable se pueda usar en todos los sistemas compatibles con el software

[19] **JDBC** es el acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

Java (windows, linux, mac, solaris, etc.), por ello toda la aplicación esta implementada con tecnología java.

3.2.2. GENERACIÓN DINÁMICA DE PÁGINAS WEB.

Utilizando código java mediante script hace a las JSPs[20] como parte de la tecnología java, la alternativa para satisfacer el uso de generación de contenido dinámico; La integración de clases de java (.class), hace posible la separación de la lógica del negocio y la presentación de la información.

3.2.3. SEPARACIÓN DE LA LÓGICA EN CAPAS.

Las ventajas que la separación de la lógica en capas ofrece están: aplicaciones más robustas debido al encapsulamiento, mantenimiento y soporte más sencillo, mayor flexibilidad, alta escalabilidad, entre otras. La aplicación web esta implementada en capas: Presentación, Negocio, Dominio, Datos y motor de persistencia.

3.2.4. USO DE COMPONENTES.

Proporcionan la infraestructura necesaria y la fontanería relacionada que permite a las aplicaciones web operar en un entorno complejo, multiplataforma y con capacidades de computación distribuida, tanto interna como externamente según se requiera en cada caso. Por ejemplo los componentes Java Beans[21] de Sun Microsystems.

---

[20] **JavaServer Pages** (**JSP**) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

[21] Los **JavaBeans** son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.

3.2.5. FACILIDAD DE ADMINISTRACIÓN Y USO.

Haciendo uso de la representación del lenguaje visual una interacción amigable entre usuario y la aplicación.

3.2.6. INDEPENDENCIA DE BASE DE DATOS.

El motor de persistencia traduce entre los dos formatos de datos: de registros a objetos y de objetos a registros, haciendo uso de este traductor el gestor de base de datos se hace independiente de la aplicación. La aplicación esta implementada con el gestor de base datos MySQL, pero con la integración del motor de persistencia, la aplicación se acopla a cualquier gestor de bases de datos.

3.2.7. CÓDIGO ABIERTO (OPEN SOURCE).

Importante respaldo de la sólida tecnología Java, donde la evolución de la aplicación se hace de manera rápida. Cualquier usuario programador puede tomar el código revisarlo y hacerle mejoras, incorporando nuevas funcionalidades o modificar las ya implementadas y con ello actualizar la versión del mismo.

3.3. ESQUEMA DEL SITIO WEB.

A continuación se presenta la Figura 2 del sitio web y las entidades que implementa describiendo cada uno de los componentes.

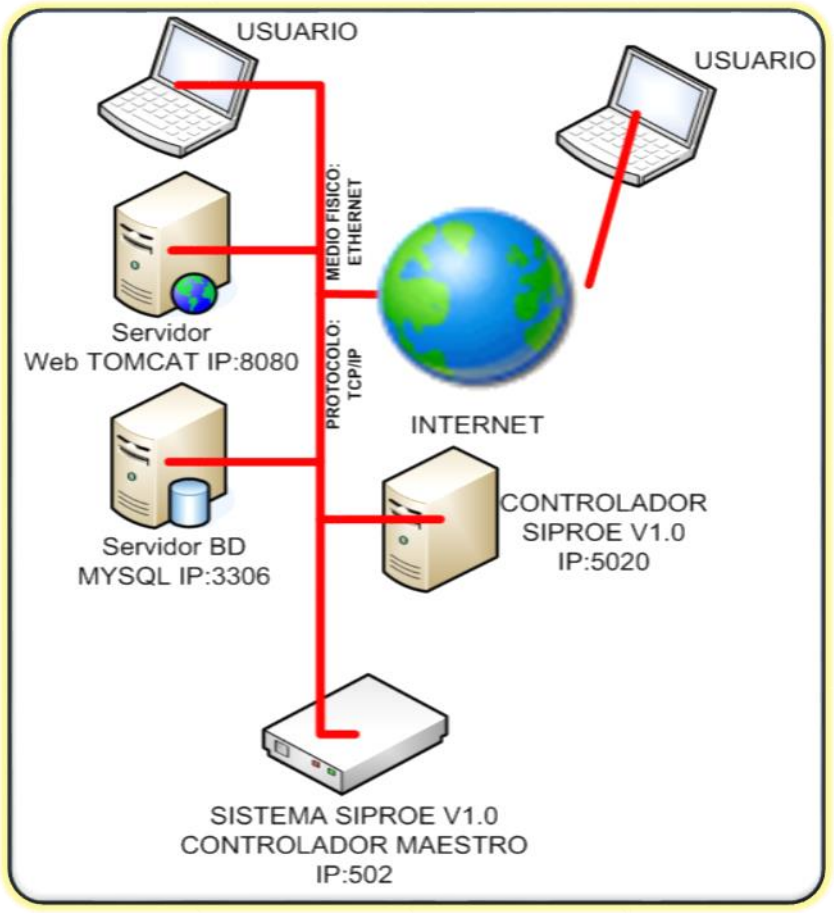

Figura 3.2: Esquema del sitio web.
Fuente [10]

3.3.1. EL CLIENTE.

Aunque el cliente no forma parte de la implementación como tal, es necesario tomarlo en cuenta ya que es el agente a quien está destinado el uso del sistema. Se entenderá como cliente al navegador web que direccione la aplicación con la url correspondiente a la aplicación del SIPROE, así en un mismo equipo pueden haber varios clientes simultáneos, claro que para la aplicación es transparente que se trate de cliente en el mismo equipo, entre los navegadores que pueden servir de clientes se pueden mencionar: Mozilla, FireFox, Internet Explorer, etc.

Dos tipos de clientes son los que el sistema puede darle soporte, uno es el cliente interno que es el que se sitúa dentro de la misma red LAN ethernet donde se encuentra el servidor que contiene la aplicación, y el otro es el cliente externo, se encuentra situado fuera de la red LAN, que puede ser desde otra red LAN o Internet, siempre y cuando el cortafuegos local esté configurado para permitir clientes externos.

3.3.2. SERVIDOR DE SERVLETS (TOMCAT).

El servidor de sitios en la Web es un programa que corre como un servicio en un equipo o dispositivo electrónico. Este escucha las peticiones de acuerdo al siguiente formato de dirección de recursos url:

http://nombre_del_servidor:numero_puerto /nombre_aplicación.

El servidor Web buscará una página dentro de un grupo de estas, que son de tipo estáticas o dinámicas; de cualquier modo, siempre devolverá algún tipo de resultado html al cliente o navegador que realizó la solicitud. Este es fundamental en el desarrollo de las aplicaciones del lado del servidor que se implementa, ya que se ejecutarán en él.

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat), es un servidor web con soporte de servlets22 y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web apache.

---

[22] La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java; esta es una de las razones por la que se seleccionó para la implementación del Sitio Web de este Trabajo de Graduación.

Debidamente instalada la aplicación web SIPROEv1.0 en Tomcat, se puede acceder al sistema SIPROE a través de una interface gráfica de usuario desde la Web, así un usuario podrá hacer uso del sistema cargando una página web a través del navegador con la url: *http://IpHost:8080/SIPROEv1.0/*, en la tabla siguiente se describe el significado de cada parámetro.

Tabla 3.1: Detalle de la url http://IpHost:8080/SIPROEv1.0/
Fuente [10]

Parámetro	Descripción
IpHost	Es la dirección IP de la PC donde está instalado tomcat.
8080	Es el puerto por defecto en el cual Tomcat escucha peticiones http.
SIPROEv1.0	Es el nombre de la aplicación que navega el sitio web del sistema SIPROE.

3.3.3. SERVICIO DE BASE DE DATOS (MYSQL).

Es el repositorio de información del sistema, este servicio es usado por la aplicación web para leer información, y guardar datos que se pueden persistir de manera directa permitiendo la administración y mantenimiento del sistema. Y también es usado por el servicio controlador administrador de eventos.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Funciona sobre multiple plataforma y es de código abierto sencillo de usar y rápido. El valor agregado que mysql le da al siproe es persistir tablas de horarios de los eventos programados.

3.3.4. SERVICIO CONTROLADOR DEL SISTEMA INTELIGENTE siSiproe.

El Servicio Controlador al sistema inteligente; Es un software de aplicación de consola creado 100% en java que soporta multihilo, multiplataforma, por medio del cual es posible la comunicación de la aplicación usuario al controlador inteligente maestro.

De aquí en adelante llamaremos siSiproe al Servicio Controlador del Sistema inteligente.

3.4. COMUNICACIÓN CON EL siSiproe.

Este tiene la particularidad de ser servidor al lado de la aplicación web y cliente Modbus/tcp al lado del controlador inteligente maestro.

Para entender con claridad como interactúa siSiproe tanto con la aplicación web como el mismo controlador inteligente, se presenta a continuación el Diagrama de Secuencia.

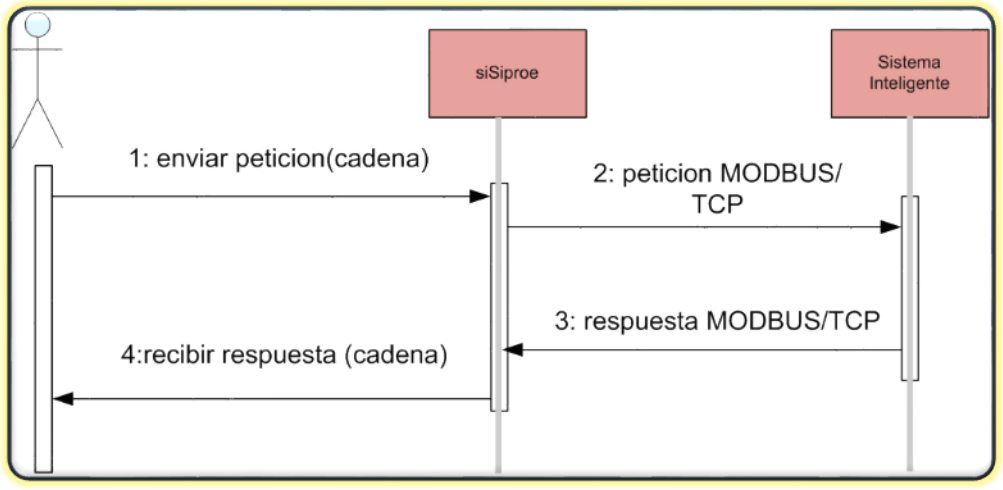

Figura 3.3: Diagrama de Secuencia de comunicación con siSiproe.
Fuente [10]

Para establecer comunicación con siSiproe, por el lado de la aplicación usuario, se abre una conexión socket a través del puerto 5020 (5020 por parecerse a 502 que es el puerto reservado de Modbus/TCP) para escuchar las peticiones de los clientes.

Cada solicitud del cliente, viene encapsulada en un formato de cadena de caracteres, ésta es transformada en petición Modbus/TCP y enviada a través de un socket por el puerto especificado en la solicitud y con la dirección IP del controlador inteligente maestro también proporcionada por la solicitud.

La solicitud tiene el formato siguiente:

Dirección IP # puerto # función # dirección de inicio # dato de función.

Tabla 3.2: Detalle del formato de la solicitud del cliente.
Fuente [10]

Parámetro	Descripción
Dirección IP	Es la dirección IP del módulo electrónico.
Puerto	Es el puerto de comunicación Modbus/TCP, su valor es 502.
función	Es la función a ejecutar y puede ser 02, 03, 05 y 16.
Dirección de inicio	Es la dirección a la cual debe aplicarse la función, debe estar soportada por el hardware de lo contrario se esperará una excepción.
Dato de función	Este campo es propio de la función.
El carácter "#"	Sirve como separador de parámetros en la cadena solicitud.

siSiproe está compuesto por un conjunto de tres clases: Siproe_SI.class, NuevoCliente.class y ModbusFunc.class.

Siproe_SI.class, es la clase principal donde se encuentra el método main() que es la que lanza el servicio quedando en estado de espera por un cliente (solicitud); cuando una nueva solicitud llega, llama a la segunda clase que a su vez, inicia un nuevo hilo que le da seguimiento a la solicitud. Cuando Siproe_SI ha levantado el

hilo de una solicitud vuelve a esperar por otra nueva. Esto hace que se pueda atender las peticiones simultáneamente (multicliente).

NuevoCliente.class, se entiende de la clase Thread de java y tiene sobrescrito el método run() donde la solicitud es capturada, separa los parámetros de la cadena con lo que se obtiene la información necesaria para construir la trama Modbus/TCP. Haciendo uso de los métodos implementados en la tercera clase se despacha la petición en el protocolo que el sistema inteligente entiende.

ModbusFunc.class, es la que hace de cliente Modbus/TCP, está compuesta por dos métodos donde se encuentran desarrolladas las funciones de Modbus/TCP (Read Input Discrete [02], read register[03], Write Single Coil [5] y write register[16]) haciendo uso de la librería del proyecto JAMOD23 y su correspondiente API. Esta clase recibe los parámetros necesarios para construir una petición Modbus/TCP y tiene implementado un cliente Modbus sobre tcp que se encarga de hacer la comunicación a través de socket con el módulo electrónico que tiene un servidor Modbus/TCP embebido.

siSiproe puede estar corriendo en el mismo servidor donde se encuentra instalada la aplicación o en un host aparte ya que toda la comunicación es realizada a través de socket.

El diagrama de clases de siSiproe es mostrado a continuación.

[23] JAMOD es una implementación orientada a objeto del protocolo Modbus hecho 100% java donde se pueden realizar fácilmente aplicaciones maestro o esclavos en vario medios de transporte (IP y serial)

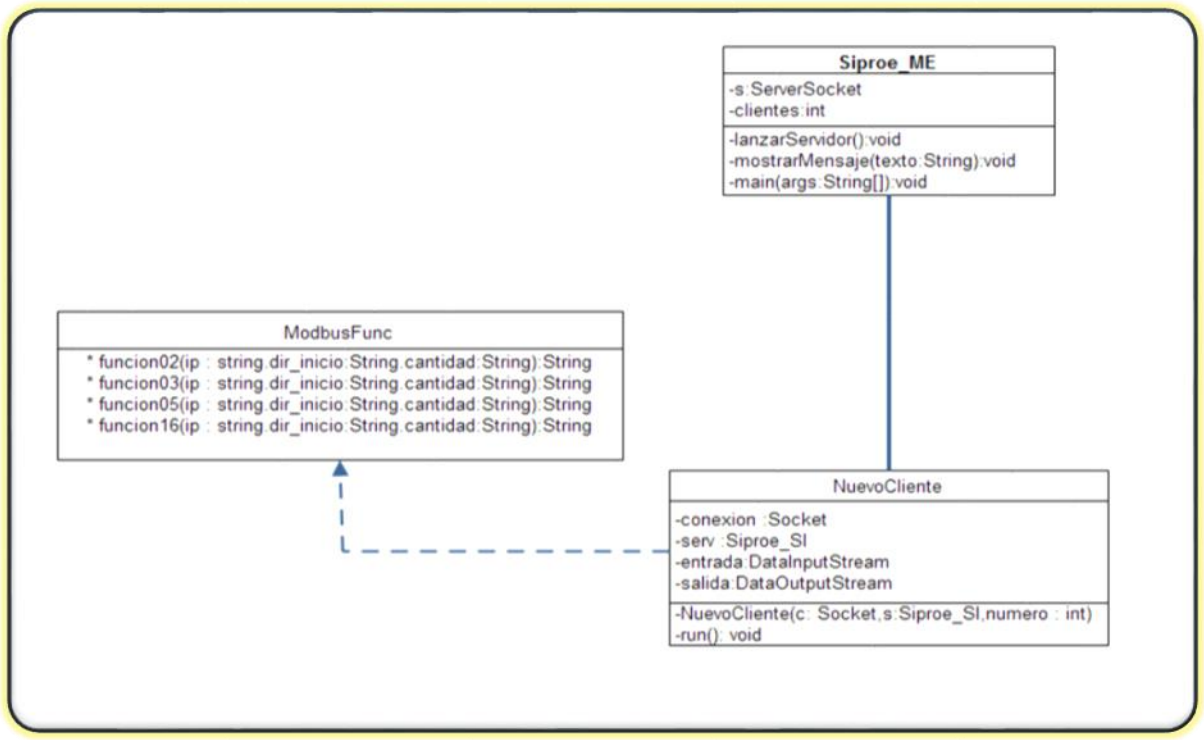

Figura 3.4: Diagrama de Clase para siSiproe.
Fuente [10]

3.5. SEPARACIÓN LÓGICA EN CAPAS.

Distintas responsabilidades no deben ser delegadas en la misma clase, y llevado esto algo más allá, en el mismo conjunto de clases. En la actualidad, la tendencia más aceptada es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas que interaccionan unas con otras a través de sus interfaces. Se trata de los sistemas denominados multicapa o n-capas.

3.5.1. ARQUITECTURA N-CAPAS.

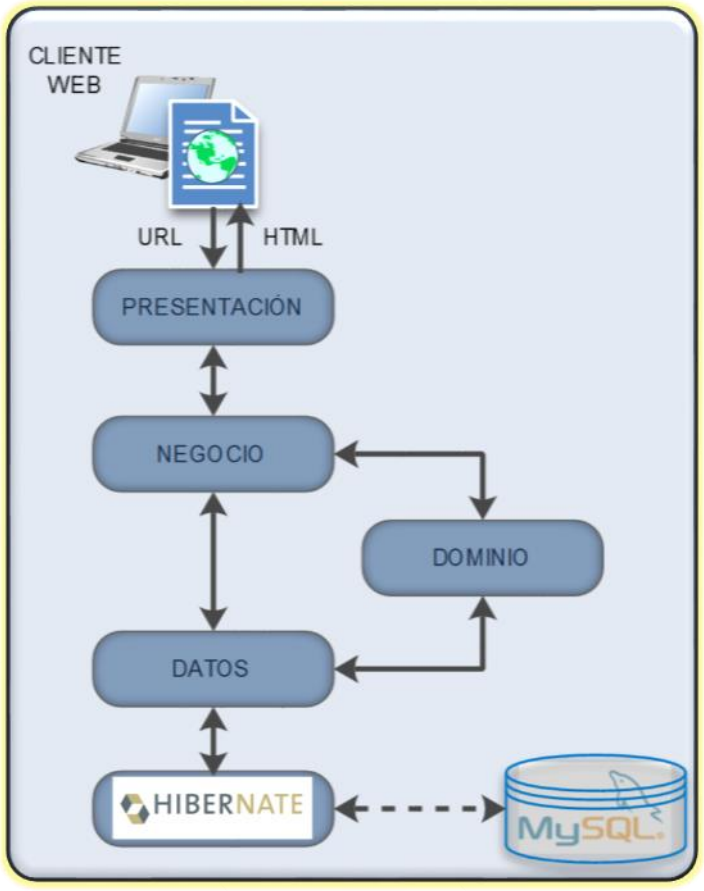

Figura 3.5: Arquitectura N-Capas.
Fuente [10]

3.5.2. CLIENTE WEB.

Esta capa es importante incluirla dentro de la arquitectura N-capas, porque es el cliente el que hace uso del sistema. Un cliente está vinculado al cada ventana del navegador en un equipo remoto o local que a través de la url, haga enlace con la aplicación SIPROE; el sistema podrá responderle al cliente en lenguaje html. Los navegadores disponibles son: Mozila, Firefox, Internet Explorer, etc.

3.5.3. PRESENTACIÓN.

Como su nombre indica, se limita a la navegabilidad y a gestionar todos aquellos aspectos relacionados con la lógica de presentación de la aplicación, como comprobación de datos de entrada, formatos de salida, internacionalización de la aplicación, autenticación, etc.

Esta capa está compuesta por páginas estáticas html y dinámicas JSPs, apoyándose Javascript, para validación de datos de entrada y de salida, además, los estilos de las páginas están a cargo de los archivos css.

3.5.4. NEGOCIO.

El resultado del análisis funcional de la aplicación viene a ser la identificación del conjunto de reglas de negocio que abstraen el problema real a tratar. Estas son las que realmente suponen el motor del sistema, dado que se basan en el funcionamiento del modelo real.

Está compuesta por clases Java (.class) donde son implementadas las políticas de uso del sistema, el mantenimiento o actualización de las clases se da sin mayor problema debido a que esta capa es independiente de las otras.

3.5.5. DOMINIO.

Todos los objetos persistentes son definidos en esta capa, son livianos y no añaden ningún tipo de carga de proceso adicional (gestión de transacciones, gestión de seguridad, control de sesiones, etc.), están compuestos por sus propiedades y métodos que permiten acceder a las ellas. Todos los objetos que la capa de datos persiste en el repositorio, son los que el recipiente de dominio contiene, por ejemplo;

Al recuperar datos de la base de datos no se puede hacer con registros ya que la aplicación entiende objetos por lo tanto se recupera con los definidos en esta capa.

3.5.6. DATOS.

Esta capa es la encargada de persistir las entidades que se manejan en negocio, que están definidas en el dominio; el acceso a los datos almacenados, la actualización, eliminación, etc. también ofrece servicios relacionados con la persistencia o recuperación de información más complejos búsqueda avanzada.

3.5.7. EL MOTOR DE PERSISTENCIA HIBERNATE.

Hibernate es una herramienta ORM24 (Object-Relational mapping) completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBoss[25] que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones. Para persistir objetos en hibernate debemos crear los archivos de mapeo, configurar hibernate, crear una fábrica de sesiones, con ella crear una sesión, acceder a los datos con la sesión controlando el manejo de datos con transacciones. Para analizar cada paso se comienza creando el archivo de mapeo.

[24] Mapeo Objeto Relacional es una técnica de programación para convertir datos entre el sistema de tipos. utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos.
[25] **JBoss** es un servidor de aplicaciones J2EE de código abierto implementado en Java puro.

3.5.7.1. ARCHIVOS DE CORRESPONDENCIA.

Los archivos de correspondencia o mapeo son archivos XML, con los que se describe DTD de hibernate que una clase es una determinada tabla y que propiedades del objeto son persistentes y a que campo de la tabla se corresponden. Además se define la relación entre los objetos, dependencias, si un objeto va a ser contenedor de otro y como se refleja en la estructuras de tablas.

Por ejemplo, el archivo de mapeo para los controladores inteligentes que son adoptados por Siproe es presentado a continuación.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
        <class name="tesis.siproe.dominio.Modulo" table="modulos" >
                <id name="id" type="long" column="id_modulo" >
                <generator class="increment"/></id>
                <property name="nombre">
                  <column name="nombre"/>
                </property>
                <property name="descripcion">
                  <column name="descripcion" />
                </property>
                <property name="ip">
                  <column name="ip"/>
                </property>
                <set name="pines" lazy="false" access="field"
            cascade ="all,delete-orphan">
            <key column="id_modulo_FK"/>
            <one-to-many class="tesis.dominio.Pines"/>
          </set>
        </class>
</hibernate-mapping>
```

Para cada clase que representa una tabla en la base de datos, un archivo de mapeo es necesario y toma una similitud como el mostrado en la, estos archivos son listados en la Tabla 3.3.

Tabla 3.3: Archivos de correspondencia de hibernate.
Fuente [10]

Archivo de Mapeo	Clase Java	Tabla MySQL
Aula.hbm.xml	Aula	aula
Departamento.hbm.xml	Departamento	departamento
EventoAnalogo.hbm.xml	EventoAnalogo	evento_analogo
EventoDigital.hbm.xml	EventoDigital	evento_digital
InputAna.hbm.xml	InputAna	input_ana
InputDig.hbm.xml	InputDig	input_dig
LlaveAcceso.hbm.xml	LlaveAcceso	llave_acceso
LogicaAnaloga.hbm.xml	LogicaAnaloga	logica_analoga
LogicaDigital.hbm.xml	LogicaDigital	logica_digital
Modulo.hbm.xml	Modulo	modulo
OutputAna.hbm.xml	OutputAna	output_ana
OutputDig.hbm.xml	OutputDig	output_dig
RelojRtc.hbm.xml	RelojRtc	reloj_rtc
Rol.hbm.xml	Rol	rol
Submodpg.hbm.xml	Submodpg	submodpg
Tarjeta.hbm.xml	Tarjeta	tarjeta
Tipo.hbm.xml	Tipo	tipo
Usuario.hbm.xml	Usuario	usuario

3.5.7.2. CONFIGURACIÓN DE HIBERNATE.

Hibernate está diseñado para operar en muchos ambientes diferentes, por lo tanto hay un número grande de parámetros de la configuración. Afortunadamente, la mayoría de ellos tienen valores por defecto sensatos.

Hibernate puede configurarse por líneas de código con lo cual si queremos hacer una modificación debemos compilar nuevamente. Esto no es muy beneficioso por lo que hibernate nos permite configurarlo con un archivo de configuración el cual puede ser XML o texto llano que su extensión es .properties.

3.5.7.3. CREAR UNA FÁBRICA DE SESIONES.

Luego de crear el objeto Configuración y setear todas las clases persistentes se debe crear una fábrica de sesiones con la cual se crean las sesiones para persistir los objetos de esta forma:

SessionFactory fabrica =cfg. buildSessionFactory();

Hibernate permite tener más de una fábrica de sesiones esto sirve para cuando la aplicación se conecta con más de una base de datos. Todo esto esta implementado en las clases java dentro de la capa de Datos.

3.5.7.4. CONECTARNOS A LA BASE DE DATOS.

Nos conectamos a una base de datos por medio de una sesión en el momento que creamos la sesión nos conectamos a la base de datos, creamos la sesión con la fábrica de sesiones ya que la sesión hereda las configuraciones que seteamos en el objeto de configuración con el cual creamos la fábrica de sesiones.

Para Crear una sesión se hace de la forma:

Session session = fabrica.openSession(); // abrir una sesión.

Cuando creamos esta sesión se disparara una excepción (error) dado que no hemos especificado todavía la base de datos donde vamos a trabajar. Para crear satisfactoriamente una sesión debemos especificar:

Tabla 3.4: Propiedades de configuración de Hibernate.
Fuente [10]

Nombre de la propiedad	Propósito
hibernate.connection.driver_class	Driver jdbc
hibernate.connection.url	Url de la base de datos
hibernate.connection.username	Usuario de la base de datos
hibernate.connection.password	Password del usuario de la base de datos
hibernate.connection.pool_size	Número máximo de conexiones concentradas

Estas propiedades se pueden configurar por medio del archivo hibernate.cfg.xml. Este archivo puede usarse como un reemplazo para el hibernate.properties o, si los dos están presentes, el archivo XML sobrescribe las propiedades. Por ejemplo. Un extracto de este archivo para la aplicación del Siproe se muestra a continuación.

```
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/siproe</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">root</property>
<property name="hibernate.connection.pool_size">100</property>
<property name="show_sql">false</property>
<property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
<property name="hibernate.hbm2ddl.auto">Update</property>
```

Debido a que se está trabajando con Mysql el driver jdbc es "com.mysql.jdbc.Driver", este es proporcionado gratuitamente y puede bajarse de internet o también se encuentra dentro del cd que se anexa a este documento.

El dialecto es especificado como "org.hibernate.dialect.MySQL" o con el compatible al servidor de la base de datos que se esté utilizando.

Los demás parámetros son fácilmente comprensibles y los cambios necesarios para adaptarse a sistemas ya implementados no requieren de mayor explicación.

3.5.7.5. LOS DIALECTOS DE SQL.

SQL trata de ser un estándar pero no lo logra, según la base de datos que estemos usando se llamaran de un modo u otro las funciones y la sintaxis no es igual en todas; por lo que se denomina escribir en un dialecto cuando escribimos SQL que solo entiende un motor de base de dato determinado. Como hibernate crea SQL en tiempo de ejecución debe saber el Dialecto que debe usar, Hibernate soporta los Dialectos que se muestran en la tabla 3.4.

3.5.7.6. ABRIR UNA SESIÓN.

Luego de haber completado toda la configuración podemos crear la sesión. La creamos de esta forma:

```
Session session = fabrica.openSession(); // abrir una sesión.
```

Abrimos una sesión dentro de ella existe un objeto encargado de controlar las transacciones llamado Transaction se debe referenciarlo para tener control sobre este objeto. Se hace de la siguiente manera:

```
Transaction tx= session.beginTransaction();
```

Al iniciar la transacción ya se está en condiciones de guardar nuestro objeto con el método de la sesión save(objeto) y se puede recuperar un objeto con el método get(objeto.class,id) donde id es el id del objeto y objeto.class es la clase del objeto. Se puede borrar con el método delete(objeto) y eliminar el objeto de la base de datos. Modificar el objeto persistente que se encuentra en la base de datos con el método upDate(objeto).

Tabla 3.5: Dialectos de hibernate.
Fuente [10]

RDBMS	Dialecto
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL with InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle 9i/10g	org.hibernate.dialect.Oracle9Dialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect

SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect
HypersonicSQL	org.hibernate.dialect.HSQLDialect
Ingres	org.hibernate.dialect.IngresDialect
Progress	org.hibernate.dialect.ProgressDialect
Mckoi SQL	org.hibernate.dialect.MckoiDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Pointbase	org.hibernate.dialect.PointbaseDialect
FrontBase	org.hibernate.dialect.FrontbaseDialect
Firebird	org.hibernate.dialect.FirebirdDialect

Luego de realizar todas las acciones que se desea solo se debe invocar el método del objeto Transaction commit() y se aplican los cambios, si surgiera algún error o las modificaciones no debieran ser aplicadas y se llama al método rollback() del objeto Transaction.

3.5.7.7. EL LENGUAJE DE INTERROGACIÓN DEL MUNDO OBJECTUAL: HQL.

El HQL (Hibernate Query Language) es un lenguaje de interrogación. En el mundo relacional disponemos del SQL (Structured Query Language) que nos permite obtener información haciendo preguntas basadas en las tablas y sus columnas. El equivalente en el mundo objetual es el HQL, que nos permite hacer preguntas basadas en los objetos y sus propiedades.

Hibernate se encarga de enlazar los dos mundos el relacional con el objetual. Traduce las consultas que se hacen desde el mundo objetual en HQL al lenguaje de interrogación del mundo relacional, el SQL, y transforma los resultados obtenidos en el mundo relacional (filas y columnas) en aquello que tiene sentido en el mundo objetual: objetos.

3.5.7.8. EJECUCIÓN DE CONSULTAS.

Existen diversos métodos para ejecutar consultas.

El método "Session.find()"

Este devuelve el resultado de la consulta en una java.util.List. Es bastante práctico si se devuelven pocos resultados, ya que los tiene que mantener en memoria.

El método "Session.iterate()"

Este método devuelve un java.util.Iterator y es práctico si la consulta nos proporciona un gran número de resultados. El iterador se encarga de cargar los objetos resultantes de la consulta uno a uno, a media que los vamos pidiendo.

La interface "Query"

La interface Query nos permite ejecutar consultas pero aporta algunas ventajas:

- Podemos especificar el número máximo de registros que queremos que se nos devuelvan.
- Podemos especificar el primer registro que queremos obtener
- Permite el uso de parámetros con nombre

3.6. LA INTERFAZ DE USUARIO.

Cuando un usuario accede al sitio web a través de un navegador de páginas web, la página que visualiza es la de inicio, desde la cual se puede apreciar las mediciones de energía activa, reactiva y aparente que se han realizado, además se puede revisar los voltajes y corrientes.

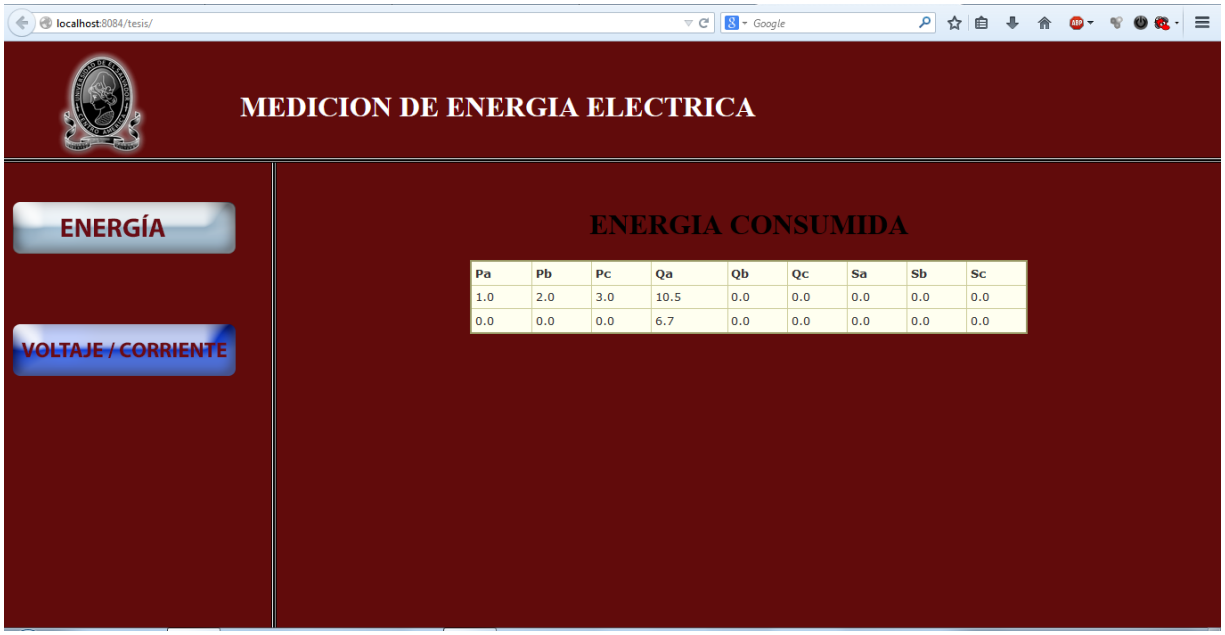

Figura 3.6: Página de inicio del sitio web.
Fuente [Autor]

RECOMENDACIONES.

- Implementar un circuito de aislamiento desde los canales analógicos de voltaje y de corriente, para proteger de una falla en la red eléctrica, todos los componentes electrónicos que conforman el medidor.

- En la programación de los microcontroladores es conveniente utilizar macros con directivas que permitan decidir si el código fuente de un macro se agrega al código principal del microcontrolador o no.

- Utilizar archivos independientes para definir variables, constantes y macros; para depurar de manera más fácil el código fuente del programa principal.

- Debido a la necesidad de mantener energizado todo el sistema del medidor, se necesita implementar un sistema de respaldo que mantenga alimentado al medidor durante un tiempo determinado, cuando existan fallas en la red; para asegurar el buen funcionamiento del medidor, y principalmente los datos almacenados en memoria.

- Uno de los dispositivos más importantes en la medición de energía eléctrica es el sensor de corriente. De la escogencia de un buen sensor depende los errores que se puedan presentar en las variables medidas

CONCLUSIONES.

- Cuando se transmite información a través de una interfaz serial SPI, no basta con habilitar el módulo de comunicación del periférico con el que se desea establecer la comunicación; hay que tomar en cuenta además, que el periférico debe de estar listo para recibir, cada vez que transmite el microcontrolador maestro, ya que si esto no se cumple los bytes enviados por el maestro se perderán o no serán leídos por el periférico. Si el periférico desea enviar uno o más bytes al microcontrolador maestro, el periférico debe de esperar a que este inicie una transmisión de datos.

- Utilizar los medios de comunicación proporcionados por la tecnología actual como internet, es una herramienta muy útil para dar soluciones ingenieriles a problemas como el manejo, control y administración de sistemas eléctricos.

- Java es una herramienta muy potente para desarrollar software independiente de la plataforma del sistema operativo.

- El uso de un software motor de persistencia hace posible la independencia del gestor de base de datos.

- El uso de código abierto permite la implementación de aplicaciones ingenieriles de bajo costo.

- El protocolo CAN permite optimizar la comunicación entre controladores, dando la facilidad de agregar más dispositivos al bus, sin que este pierda su rendimiento.

- El protocolo industrial MODBUS permite manejar dispositivos remotos; independientes del medio físico que se implemente para hacer peticiones, sea Ethernet, RS232 o CAN.

- Para la implementación de un medidor electrónico de energía eléctrica en sistemas trifásicos, se recomienda el uso del circuito integrado ADE7758 (ANALOG DEVICES) porque cuenta con la ventaja que tiene registros de potencia aparente, lo cual permite ver el comportamiento del circuito para cargas no lineales y sistemas no balanceados.

- Otra ventaja del ADE7758 son registros configurables en el mismo, lo que permite entre otras opciones remover problemas de offset y desfase entre los canales de corriente y tensión. A su vez el ADE7758 tiene un registro de interrupción el cual indica si existe sobretensión, sobrecorriente, caída de tensión, error de secuencia de fase entre otros.

- Una de las desventajas principales en el ADE7758 es el error en la medición de tensión y por ende en potencia aparente, cuando existen armónicos en el canal de tensión. Lo anterior se debe a que la señal de tensión pasa por un filtro paso bajas de frecuencia de corte de 160Hz y atenúa la señal antes de calcular la potencia aparente.

BIBLIOGRAFIA

[1] I. Aliaga Vargas, P. Bravo y R. Bustamante, « Scribd,» 2 Noviembre 2011. [En línea]. Available: https://www.google.com.sv/search?q=AUTOMATIZACION+DE+PROCESOS+INDUSTRIALESSENSORES+DE+CORRIENTE&ie=utf-8&oe=utf-8&rls=org.mozilla:es-ES:official&client=firefox-a&channel=fflb&gws_rd=cr&ei=t8NPVJ3zMrWCsQTdp4DgDQ#rls=org.mozilla:es-ES:official&channel=fflb&. [Último acceso: 19 Marzo 2013].

[2] L. Mattei, «electroyou,» 5 Noviembre 2011. [En línea]. Available: http://www.electroyou.it/luka889/wiki/la-bobina-di-rogowski. [Último acceso: 20 Marzo 2013].

[3] L. Betancourt, «Blogger,» 25 Mayo 2007. [En línea]. Available: http://lbetanc7230.blogspot.com/2007/05/tipos-de-sensores.html. [Último acceso: 19 Marzo 2013].

[4] D. Industry, «Direct Industry,» [En línea]. Available: http://www.directindustry.es/prod/electrohms/sensores-corriente-efecto-hall-lazo-abierto-abribles-54132-367917.html. [Último acceso: 19 Marzo 2013].

[5] Tectronix, «Tectronix,» [En línea]. Available: http://www.tectronix.cl/sensor-de-corriente-acs715.html. [Último acceso: 19 Marzo 2013].

[6] W. Koon, «Analog Divices,» [En línea]. Available: http://www.analog.com/static/imported-files/tech_articles/16792408482720MI_Issue3_2001_pg52-53_analog_Spanish.pdf. [Último acceso: 19 Marzo 2013].

[7] A. Harney, «Analog Dialogue,» Enero 2009. [En línea]. Available: http://www.analog.com/library/analogdialogue/archives/43-01/smart_metering.pdf. [Último acceso: 27 Marzo 2013].

[8] I. Analog Devices, «Analog Device,» Octubre 2011. [En línea]. Available: http://www.analog.com/static/imported-files/data_sheets/ADE7758.pdf. [Último acceso: 13 Febrero 2013].

[9] C. Miranda Estepa y J. Ronquillo Guererro, «Diseño y construcción de bus de datos y sensores para las prácticas de NACC,» Universidad Politecnica de Cataluña, Cataluña, España, 2008.

[10] E. A. Castellon Torres y C. R. Romero Miranda, «DISEÑO Y CONSTRUCCION DE UN SISTEMA INTELIGENTE PROGRAMABLE VIA ETHERNET PARA LA OPTIMIZACION DE ENERGIA ELECTRICA DESDE UNA APLICACIÓN WEB,» UES, San Salvador, El Salvador, 2009.

[11] A. Ramis Fuambuena, «Página Personal de ALBERTO RAMIS FUAMBUENA,» 2012. [En línea]. Available: http://personales.alumno.upv.es/alrafua/asignaturas/SES/Buses/CAN/can.html. [Último acceso: 17 Mayo 2013].

[12] MikroElektronika, «MikroElektronika,». http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/. [Último acceso: 20 Mayo 2013].

[13] «DISPOSITIVOS LÓGICOS MICROPROGRAMABLES,» 4 Enero 2013. [En línea]. Available: http://perso.wanadoo.es/pictob/microcr.htm. [Último acceso: 20 Mayo 2013].

[14] M. T. Inc., «Microchip,» 2007. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39761b.pdf. [Último acceso: 14 Febrero 2013].

[15] Anónimo, «Wikipedia,» 13 Noviembre 2013. [En línea]. Available: http://es.wikipedia.org/wiki/Secure_Digital. [Último acceso: 29 Marzo 2013].

[16 Suky, «Todopic,» 2 Octubre 2009. [En línea]. Available: http://www.todopic.com.ar/foros/index.php?topic=27786.0.

] [Último acceso: 4 Junio 2013].

[17] GEDA, «Sakai,» 2005. [En línea]. Available: http://ik.itba.edu.ar/~jacoby/Lab%20micros%202013/Teoricas/Clase8%20-%20SD%20Card,%20IIC%20%28Secure%20Digital%20Card,%20Inter%20Integrated%20Circuit%20Bus%29/CLASE8%20-%20SD/Info%20SD/Funcionamiento%20de%20la%20tarjeta%20SD-MMC.pdf. [Último acceso: 12 Julio 2013].

[18] A. Rossini, «arrrosini.com.ar,» 14 Agosto 2011. [En línea]. Available: http://www.arossini.com.ar/. [Último acceso: 4 Junio 2013].

[19] B. Y. Electronics, «YHDC,» [En línea]. Available: http://www.yhdc.com/en/product/351/. [Último acceso: 27 Marzo 2013].

[20] AKENAFAB, «todopic,» 29 Julio 2009. [En línea]. Available: http://www.todopic.com.ar/foros/index.php?topic=27011.0.

[21] C. C. Services, «CCS Inc.,» Septiembre 2013. [En línea]. Available: http://www.ccsinfo.com/downloads/ccs_c_manual.pdf. [Último acceso: 20 Nobiembre 2013].

[22] M. T. Inc., «Microchip,» Noviembre 2011. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/25074a.pdf. [Último acceso: 16 Abril 2013].

ANEXOS.

A. PROPOSITO DE ESTA GUIA.

El propósito de esta guía es proporcionar a los usuarios del SIPROE una referencia rápida para poder implementar el sistema cuando se requiera, esta incluye las configuraciones en hardware y software

A.1. RESUMEN DE SECCIONES.

Este manual se divide en varias secciones las cuales se resumen en la siguiente:

Tabla A.1: Secciones del manual de usuario
Fuente [10]

SECCION	PROPOSITO
Configuración del controlador inteligente maestro	Provee una guía para configurar el controlador maestro
Configuración del controlador inteligente de reloj de tiempo real	Provee la forma en que se configura el reloj de tiempo real
Configuración del controlador de Acceso	Provee la forma de configurar el controlador de acceso
Configuración del controlador inteligente de propósito general	Provee la manera en que se configura el controlador de propósito general
Instalación de aplicación SIPROE en Tomcat	Provee una guía para la puesta en marcha de los servicios necesarios para el funcionamiento de SIPROE
Interfaz gráfica de la aplicación web	Provee una descripción de la interfaz gráfica de la aplicación web de SIPROE

A.1.1. INFORMACION ADICIONAL.

Todo el firmware, el software, hoja de especificaciones de componentes y los programas necesarios para la implementación del sistema inteligente se encuentran en el CD anexo al libro, esto para facilitar su implementación.

A.1.2. CONFIGURACION DEL CONTROLADOR INTELIGENTE MAESTRO.

El servidor embebido en el controlador inteligente maestro usa el protocolo TCP/IP para comunicarse en la red ethernet, soporta ARP, UDP, TCP, ICMP, Telnet, TFTP, DHCP, y SNMP. Toda la información acerca del servidor Modbus/TCP se encuentra en el CD adjunto a este trabajo de graduación. Es necesario configurarlo manualmente para personalizar algunos parámetros y ajustarlo a la aplicación SIPROE; para hacerlo es necesario usar el modo de configuración que el servidor tiene disponible de fábrica.

Se hace a través de una conexión TELNET sobre la red ethernet; toda la información es guardada en la memoria no volátil que posee el módulo Xport.

A.1.2.1. BUSCANDO UN CONTROLADOR INTELIGENTE MAESTRO EN LA RED.

Cuando existe un nuevo controlador inteligente maestro es necesario configurarle algunos parámetros, pero para ello se necesita saber que IP tiene asignado en ese preciso momento. La clave es revisar el hardware en el controlador maestro, ahí está embebido el dispositivo Xport y en la etiqueta tiene impreso su dirección MAC.

El Xport ocupado en este Trabajo de Graduación tiene la dirección MAC.

MAC: 00-20-4A-8F-64-5A

Se tiene dos opciones para satisfacer el problema.

- Utilizando el software *Device Installer* de Latronix (disponible en el CD adjunto), leer la documentación para el uso del mismo.

- Ocupando un analizador de protocolos en redes Ethernet, tal es el caso de Ethereal solo basta capturar en modo promiscuo y revisar que IP le corresponde el dispositivo con la dirección MAC que se está buscando.

A.1.2.2. ACCESANDO A MODO CONFIGURACIÓN.

Para configurar el controlador maestro sobre la red es necesario establecer una conexión telnet por el puerto 9999.

Para establecer una conexión telnet.

1. Del menú Inicio de Windows, hacer click en *Ejecutar* y escriba el siguiente comando, donde x.x.x.x es la dirección IP y 9999 es el puerto fijo de configuración del módulo electrónico.

```
Windows: telnet x.x.x.x 9999
UNIX:    telnet x.x.x.x:9999
```

La IP que en este trabajo de graduación se ocupó es IP: 192.168.1.50

2. Hacer Click en *Ok,* la siguiente información aparecerá en pantalla.

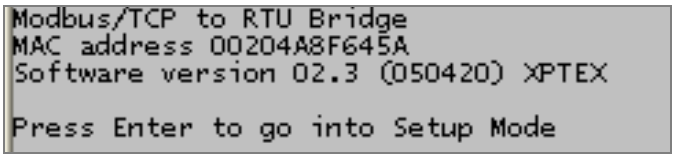

Figura A.1: Pantalla de inicio en Modo Configuración del controlador maestro. Fuente [10]

3. Para entrar en modo configuración, presione *Enter* antes de 5 segundos. Aquí se cambian los parámetros que se necesitan personalizar.

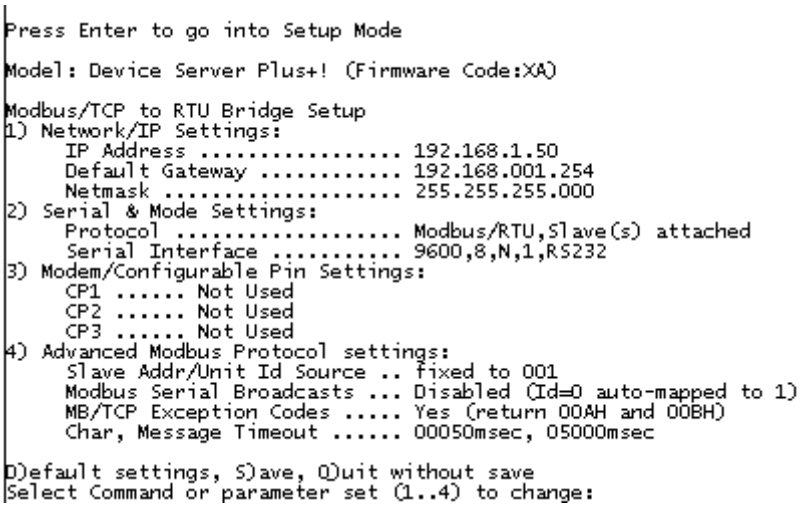

Figura A.2: Pantalla de Opciones en Modo Configuración del controlador maestro. Fuente [10]

A.1.2.3. CONFIGURACIÓN IP DEL SERVIDOR MODBUS/TCP.

Seleccione 1 para configurar los parámetros de red del controlador maestro y siga las instrucciones de la pantalla.

La IP debe ser un valor único dentro de la red. Si la IP que se le asigne ya está siendo ocupada por otro dispositivo se presenta en los leds un código de Error que puede ser consultado en la guía de usuario del Xport en el CD adjunto. *No debe ser activado en modo DHCP*, puede causar fallo de conexión, porque en la aplicación siproe debe poseer IP fija.

Los demás parámetros (2…4) afortunadamente no son necesarios modificarlos. En caso de requerir modificaciones en algún parámetro específico, refiérase a la documentación para el Xport en el CD adjunto. El último paso es guardar cambios y salir.

A.1.2.4. SALIR GUARDANDO LA CONFIGURACIÓN.

Presione la tecla S y automáticamente el módulo guarda la configuración en la memoria no volátil y se reinicia.

El Controlador maestro esta ahora configurado y listo para que SIPROE funcione satisfactoriamente.

A.2. CONFIGURACIÓN CAN.

Setup Criteria	
Oscilator Frequency	10.000 MHz
Target CAN Bus Baud Rate	125.000 kbps

Selected Options	
BRP-1 (Baud Rate Prescaler)	4
Tq (Time Quanta)	1.000 µs
Number of Time Quanta	8
% Error of Target Baud Rate	0.0 %

Bit Timing Setup in Tq	
Propagation Delay	1
Phase Segment 1	3
Phase Segment 2	3
Syncronization Jump Width (SJW)	1

Multiple bit sampling is off. Wakeup filter is off.

Bit Timing Diagram

Configuration Register Setup (PIC18/MCP251X) (neoVI blue/green, ValueCAN 2)		
Register	Binary	Hexadecimal
CNF1/BRGCON1	b'00000100'	0x04
CNF2/BRGCON2	b'10010000'	0x90
CNF3/BRGCON3	b'00000010'	0x02

A.3. PRESUPUESTO DE MATERIALES.

La lista de los componentes que forman parte del medidor de energia y el costo de los mismos se muestra en el siguiente cuadro resumen.

Tabla A.2: Lista de componentes y precios.

CANTIDAD	DESCRIPCION	IDENTIFICADOR	PRECIO UNITARIO	SUB-TOTAL
12	RESISTENCIA 2.2 OHMS X 0.25W	R5, R6 R7, R8, R13, R14, R15, R16, R21, R22, R23, R24	0.1	1.2
10	RESISTENCIA 100 OHMS X 0.25W	R10, R12, R18, R20, R26, R28	0.1	1
1	RESISTENCIA 120 OHMS X 0.25W	R55	0.1	0.1
1	RESISTENCIA 200 OHMS X 0.5W	R48	0.1	0.1
1	RESISTENCIA 240 OHMS X 0.25W	R3	0.19	0.19
1	RESISTENCIA 270 OHMS X 0.25W	R1	0.19	0.19
1	RESISTENCIA 330 OHMS X 0.25W	R1	0.23	0.23
1	RESISTENCIA 390 OHMS X 0.25W	R4	0.19	0.19
1	RESISTENCIA 820 OHMS X 0.25W	R2	0.23	0.23
4	RESISTENCIA 1K OHMS X 0.25W	R40, R41, R42, R59	0.1	0.4
10	RESISTENCIA 1.5K OHMS X 0.5W	R9, R11, R17, R19, R25, R27, R29, R30, R33, R36, R38	0.1	1
4	RESISTENCIA 1.8K OHMS X 0.25W	R43, R44, R45, R60	0.1	0.4
2	RESISTENCIA 4.7K OHMS X 0.25W	R52, R53	0.1	0.2
8	RESISTENCIA 10K OHMS X 0.25W	R46, R47, R49, R50, R54, R56, R57, R58	0.23	1.84
3	RESISTENCIA 2.2M OHMS X 0.5W	R29, R32, R35	0.1	0.3
4	CAPACITOR CERAMICO 22pF	C17, C18, C19, C20	0.2	0.8
10	CAPACITOR CERAMICO 10nF	C5,C6, C7, C8, C9, C10, C11, C12, C13, C14	0.35	3.5
9	CAPACITOR CERAMICO 100nF	C1, C2, C15, C21, C22, C27, C28, C29, C30	0.35	3.15
3	CAPACITOR ELECTROLITICO	C3, C4, C16	0.19	0.57

	1uF X 50V			
4	CAPACITOR ELECTROLITICO 10uF X 16V	C23, C24, C25, C26	0.2	0.8
3	LED VERDE 12V	D2, D4, D6	0.5	1.5
2	LED AZUL 12V	D3, D7	0.5	1
1	LED RGB	D5	1.08	1.08
1	DIODO 1N4148	D1	0.25	0.25
1	DIODO 1N5401	D8	0.25	0.25
1	MICROCONTROLADOR PIC18F4685	U3	7.37	7.37
1	IC ADE7758	U4	8.31	8.31
1	IC MCP2551	U8	1.02	1.02
1	IC MAX232	U6	3.8	3.8
1	IC DS1307	U7	2.65	2.65
1	IC 74LS164	U5	1.67	1.67
1	FUSIBLE CON BASE	FU1	0.41	0.41
1	SWITCH	SW1	0.74	0.74
1	JACK DC	J2	0.37	0.37
1	TERMINAL DB9 HEMBRA	J5	0.56	0.56
2	TRIMMER 5K OHMS	CONTRAST, LIGHT	1.75	3.5
2	IC LM317	U1, U2	1	2
1	BUTTON	SW2	0.25	0.25
3	TBLOCK-M2	J3, J4, IA, IB, IC,	0.5	1.5
2	CRISTAL 10 MHZ	X1, X2	0.75	1.5
1	CRISTAL 32.768 KHZ	X3	1.4	1.4
1	BASE DIL 40 PINES		0.3	0.3
1	BASE DIL 28 PINES		0.3	0.3
1	BASE DIL 14 PINES		0.15	0.15
1	BASE DIL 16 PINES		0.19	0.19
2	BASE DIL 8 PINES		0.25	0.5
2	DISIPADORES DE 2 ALAS		0.11	0.22
1	ADAPTADOR DB9 A RJ45		1.27	1.27
1	TABLETA DE FIBRA DE VIDRIO 6"X10"		11.37	11.37
4	CABLE AWG16 ROJO		0.3	1.2
4	CABLE AWG16 NEGRO		0.3	1.2
3	CABLE AWG16 AZUL		0.3	0.9
3	CABLE AWG16 BLANCO		0.3	0.9
3	TRANSFORMADOR DE CORRIENTE		14.98	44.94
1	PANTALLA LCD	LCD1	4.47	4.47
TOTAL				**125.43**

B. GUIA DE USUARIO MEDIDOR DE ENERGIA.

Paso 1: Conecte los terminales en el medidor de energía, luego conecte las pinzar de muestreo de voltaje y los transformadores de corriente.

Paso 2: Encienda el equipo y espere un par de segundos, si todo está bien el indicador cuadrado se iluminará de verde, de lo contrario será rojo indicando un error y mostrara un código de error en el LCD que puede ser alguno de los que se muestran en la tabla siguiente.

Tabla B.1: Descripción de los códigos de error.

CODIGO	DESCRIPCION
ERROR 01	La ranura SD se encuentra vacía.
ERROR 02	Memoria bloqueada, no se puede escribir.
ERROR 04	Error al iniciar el modo SPI.
ERROR 08	No se puede implementar el sistema FAT

Paso 3: Si necesita configurar la fecha y hora del módulo RTC, debe conectarse al terminal DB9 y configurarlo a una velocidad de 115200 baudios, también por medio de la interface serial puede tener una mejor idea del error que ha encontrado el sistema.

C. CODIGO PRINCIPAL.

A continuación se muestra el código principal usado en este proyecto, las librerías usadas se encuentra con el CD adjunto a este documento.

```
#include "C:\Users\OTONIEL\Desktop\TESIS\main.h"
#include <stdlib.h>
#include <string.h>
#include "configurar_spi.c"
#include "ds1307.c"
#include "flex_lcd_3pins.c"
#include "Timestamp.c"
#include "FAT16.c"
#include "ADE7758.c"

char LFN[11]="ENERGY.CSV";
char SFN[11]="ENERGY.CSV";
char LFN2[14]="VOLT_CORR.CSV";
char SFN2[14]="VOLT_CORR.CSV";
char msj[20],timestamp[20];
char Buff1[512];
int16 y;
int8 cmd=0;
int8 cont=0,desv=0, minuto=0;
int8 error_code = 0; // variable utilizada para identificar errores
unsigned int32 IRQ;
float32 Va,Vb,Vc,Ia,Ib,Ic,Pa,Pb,Pc,Qa,Qb,Qc,Sa,Sb,Sc;


void mostrar(int);
void clearString(char*);
void rcct(void);
int1 sd_status(void);
void preparar_almacenamiento();
float validarV(float);
float validarI(float);

#int_TIMER3
void  TIMER3_isr(void)
{
   cont++;
   if(cont==19)
      {
      ds1307_get_time(HH,MM,SS);
      ds1307_get_date(D,M,A,DDS);
      Va = validarV(getVRMS(PHASE_A)*Fv);
      Ia = validarI(getIRMS(PHASE_A)*Fi);
      Vb = validarV(getVRMS(PHASE_B)*Fv);
      Ib = validarI(getIRMS(PHASE_B)*Fi);
      Vc = validarV(getVRMS(PHASE_C)*Fv);
      Ic = validarI(getIRMS(PHASE_C)*Fi);
      y=A+2000;
      sprintf(timestamp,"%Lu-%02u-%02u %02u:%02u:%02u\0",y,M,D,HH,MM,SS);

      clearString(Buff1);

sprintf(Buff1,"%s,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f\r\n",timestamp,Va,Vb,Vc,Ia,Ib,
Ic);
      if (FAT_OpenAddFile(SFN2,0,Buff1)==0)
```

```
         printf("\r\nError escribiendo en archivo \"%s\".\r\n",SFN);

      cont=0;
      minuto++;
      if(minuto==5)// mediciones cada 5 minutos
         {
            Pa += getWattHR(PHASE_A);
            Pb += getWattHR(PHASE_B);
            Pc += getWattHR(PHASE_C);
            Qa += getVARHR(PHASE_A);
            Qb += getVARHR(PHASE_B);
            Qc += getVARHR(PHASE_C);
            Sa += getVAHR(PHASE_A);
            Sb += getVAHR(PHASE_B);
            Sc += getVAHR(PHASE_C);
            clearString(Buff1);

sprintf(Buff1,"%s,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f,%3.4f\r\n",timesta
mp,Pa,Pb,Pc,Qa,Qb,Qc,Sa,Sb,Sc);
               if (FAT_OpenAddFile(SFN,0,Buff1)==0)
                  printf("\r\nError escribiendo en archivo \"%s\".\r\n",SFN);
            Pa=0;
            Pb=0;
            Pc=0;
            Qa=0;
            Qb=0;
            Qc=0;
            Sa=0;
            Sb=0;
            Sc=0;
            minuto=0;
         }
      }

}

#int_RDA
void  RDA_isr(void)
{
   clearString(msj);
   gets(msj);
   rcct();
}



void main()
{
   int8 j;
   Va=0;
   Ia=0;
   Vb=0;
   Ib=0;
   Vc=0;
   Ic=0;
   Pa=0;
   Pb=0;
   Pc=0;
   Qa=0;
   Qb=0;
   Qc=0;
   Sa=0;
   Sb=0;
```

```
   Sc=0;
   ds1307_init();
   lcd_init();
   mostrar(2);
   delay_ms(2000);
   preparar_almacenamiento();
   if(error_code == 0)
      output_low(OK);
   else
      output_low(ERROR);

    //Se verifica que en la SD card se encuetre el archivo ENERGY.CSV, sino se crea
   if (FAT_FindFile(SFN,DirectorioRaiz) == 0)
   {
      Buff1="Time,Pa,Pb,Pc,Qa,Qb,Qc,Sa,Sb,Sc\r\n";
      if(FAT_CreateFile(LFN,SFN,0,Buff1)==0)
         printf(" Error creando archivo \"%s\".\r\n",SFN);
      else
         printf("\r\n Archivo \"%s\" creado.\r\n",SFN);
   }
   else
      printf("\r\n Archivo %s de trabajo encontrado",SFN);


   //Se verifica que en la SD card se encuetre el archivo VOLT_CORR.CSV, sino se
crea
   if (FAT_FindFile(SFN2,DirectorioRaiz) == 0)
   {
      Buff1="Time,Va,Vb,Vc,Ia,Ib,Ic,\r\n";
      if(FAT_CreateFile(LFN2,SFN2,0,Buff1)==0)
         printf(" Error creando archivo \"%s\".\r\n",SFN2);
      else
         printf("\r\n Archivo \"%s\" creado.\r\n",SFN2);
   }
   else
      printf("\r\n Archivo %s de trabajo encontrado",SFN2);

   ADE7758_INIT();   // Se inicializa la operacion del ADE7758
   configurarADE7758();
   getWattHR(PHASE_A); //se ignoran las primeras lecturas, para evitar algun error
al inicio
   getWattHR(PHASE_B);
   getWattHR(PHASE_B);
   getVAHR(PHASE_A);
   getVAHR(PHASE_B);
   getVAHR(PHASE_C);
   getVARHR(PHASE_A);
   getVARHR(PHASE_B);
   getVARHR(PHASE_C);
   IRMS(PHASE_A);
   IRMS(PHASE_B);
   IRMS(PHASE_C);
   VRMS(PHASE_A);
   VRMS(PHASE_B);
   VRMS(PHASE_C);
   setup_timer_3(T3_INTERNAL|T3_DIV_BY_8);
   enable_interrupts(INT_TIMER3);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);

   while(TRUE)
   {
     IRQ=getResetInterruptStatus();
```

```
      if(bit_test(IRQ,AEHF))   // Algun buffer WATTHR medio lleno.
         {
         Pa += getWattHR(PHASE_A);
         Pb += getWattHR(PHASE_B);
         Pc += getWattHR(PHASE_C);
         }
       else if(bit_test(IRQ,REHF))   // Algun buffer VARHR medio lleno.
         {
         Qa += getVARHR(PHASE_A);
         Qb += getVARHR(PHASE_B);
         Qc += getVARHR(PHASE_C);
         }
       else if(bit_test(IRQ,VAEHF))   // Algun buffer VAHR medio lleno.
         {
         Sa += getVAHR(PHASE_A);
         Sb += getVAHR(PHASE_B);
         Sc += getVAHR(PHASE_C);
         }
      else if(bit_test(IRQ,ZXA))
         {
         Va=getVRMS(PHASE_A);
         Ia=getIRMS(PHASE_A);
         }
      else if(bit_test(IRQ,ZXB))
         {
         Vb=getVRMS(PHASE_B);
         Ib=getIRMS(PHASE_B);
         }
      else if(bit_test(IRQ,ZXC))
         {
         Vc=getVRMS(PHASE_C);
         Ic=getIRMS(PHASE_C);
         }
      else if(bit_test(IRQ,RESET))   // Reinicio del ADE7758 por software.
         {
         ADE7758_INIT();
         }

   }
}

/*********************************************************************************
*********
* Esta funcion compruba la existencia fisica de la tarjeta SD y que no este
bloqueada contra
* escritura, tambiÚn inicia el modo SPI y el sistema FAT en la tarjeta.
*********************************************************************************
********/
void preparar_almacenamiento(void)
{
   // Comprueba la presencia de la memoria SD y que este desbloqueada para escritura
   int j;
   for(j=0;j<50;j++)
   {
      if(sd_status())
         break;
   }
   // Se inicializa la memoria SD en el modo SPI ******************************
   for(j=0;j<50;j++)
      {
      if(SDCard_Init())
         {
         bit_clear(error_code,2);
```

```
         printf("\r\n SD card en modo SPI");
         break;
         }
      else
         {
         if(!bit_test(error_code,2))
            {
            printf("\r\n SD card: error al iniciar modo SPI.");
            bit_set(error_code,2);
            }
         }
      }

   // Se inicia el sistema FAT para el almacenamiento de la informacion ********
   for(j=0;j<50;j++)
   {
      if(Fat_Init())
         {
         printf("\r\n Sistema FAT iniciado");
         bit_clear(error_code,3);
         break;
         }
      else
         {
         if(!bit_test(error_code,3))
            {
            printf("\r\n No se puede implementar sistema FAT");
            bit_set(error_code,3);
            }
         }
   }
}

//*****************************************************************************
**
   void mostrar(int opc)
{  switch(opc)
   {  case 1:
puts("\r\n_____________________________________________________________________");
               puts("Ingrese los datos de fecha y hora en formato timestamp");
               puts("Es decir, un string o cadena del tipo aaaa-mm-dd hh:mm:ss ");
               puts("o sea Anio-Mes-Dia Hora:Minuto:Segundo.");

puts("_____________________________________________________________________");
               break;

      case 2:  ds1307_get_time(HH,MM,SS);
               ds1307_get_date(D,M,A,DDS);
               Y = (int16) (A+2000);
               printf("\r\n***********************************\r\n");
               printf("* Fecha: %02u-%02u-%Lu\r\n",D,M,Y);
               printf("* Hora : %02u:%02u:%02u\r\n",HH,MM,SS);
               printf("*_____________________________________________\r\n");
               printf("* Comandos:\r\n");
               printf("* 1: presione 1 para configurar fecha y hora\r\n");
               printf("* 2: presione 2 para actualizar datos\r\n");
               printf("***********************************\r\n");
               printf("\r\n");
               break;
      default:
               puts("(^=^) OPCION INVALIDA\r\n");
               break;
   }
```

```
}

/*****************************************************************************
* Esta funcion de encarga de limpiar el arreglo de caracteres empleado en la   *
* en la comunicacion rs232 con el reloj calendario, con el objeto de evitar el *
* remanente de caracteres basura que puede crear un mal comportamiento de las  *
* funciones que hacen uso de este arreglo.                                     *
*****************************************************************************/
void clearString(char* s)
{  int i;
   for(i=0; i<strlen(s); i++)
      *s[i] = '\0';
}

/*****************************************************************************
* Funcion empleada para determinar a que corresponde los caracteres ingresados *
* por el puerto RS232, si es la cadena de datos modifica el reloj calendario,  *
* sino solo envia el comando a la funcion que muestra el valor del reloj       *
* calendario atraves del modulo RS232.
*****************************************************************************/
void rcct(void)
{
if (strlen(msj)<= 2)
      {
         cmd = atoi(msj);
         mostrar(cmd);
      }
   else if(cmd == 1)
      {  decode_timestamp(msj,D,M,y,HH,MM,SS);
         A=(int8) (y-2000);
         ds1307_set_date_time(D,M,A,dw,HH,MM,SS);
         mostrar(2);
         cmd = 0;
      }
}

/*****************************************************************************
* Funcion que determina el estado de la memoria sd, si se detecta la presencia *
* fisica de la memoria y si se encuetra en modo lectura/escritura entonces     *
* devuelve el valor TRUE, FALSE en caso contrario.                             *
*****************************************************************************/
int1 sd_status(void)
{
   int1 x,y;
   x = input(SD_CD);
   y = !input(SD_CL);
   switch(x)
      {
      case TRUE:
         if(y == 1)
            {
            bit_clear(error_code,0);
            bit_clear(error_code,1);
            //output_high(ERROR);
            return TRUE;
            }
         else if((y == 0) && (bit_test(error_code,1) == 0))
            {
            printf("\r\n Memoria bloqueada, NO SE PUEDE ESCRIBIR solo leer");
            bit_set(error_code,1);
            //output_low(ERROR);
            lcd_gotoxy(1,1);
            printf(lcd_putc,"ERROR %02i",error_code);
```

```
            return FALSE;
            }
         else if((y == 0) && (bit_test(error_code,1) == 1))
            {
            //output_low(ERROR);
            lcd_gotoxy(1,1);
            printf(lcd_putc,"ERROR %02i",error_code);
            return FALSE;
            }
         break;
      case FALSE:
         if(bit_test(error_code,0)==0)
            {
            printf("\r\n La ranura SD se encuentra vacia.");
            bit_set(error_code,0);
            //output_low(ERROR);
            lcd_gotoxy(1,1);
            printf(lcd_putc,"ERROR %02u",error_code);
            return FALSE;
            }
         else
            {
            //output_low(ERROR);
            lcd_gotoxy(1,1);
            printf(lcd_putc,"ERROR %02u",error_code);
            return FALSE;
            }
         break;
      default:
         printf("\r\n Estado de la memoria sd desconocido.");
      }
}

float validarV(float x)
{
   if(x < 24.0)
      return 0.0;
   else
    return x;
}

float validarI(float x)
{
   if(x < 0.4)
      return 0.0;
   else
    return x;
}
```